VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra elektroniky a telekomunikační techniky

Mezidoménová důvěra v sítích nové generace

The interdomain trust in the next generation networks

2010 Bc. Jiří Mrkva

Čestné prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne…….. ……

Podpis diplomanta

Poděkování

Tímto chci poděkovat panu doc. Ing. Miroslavu Vozňákovi, Ph.d. za pomoc a čas, který mi věnoval a za to, že se se mnou nezištně podělil o své odborné a praktické zkušenosti.

Abstrakt

Práce je zaměřená na zabezpečení mezidoménového propojení SIP serverů. Úvodní kapitola pojednává o architektuře a protokolech v NGN sítích. Následně jsou objasněny principy nalezení cílového SIP Proxy serveru. Podstatná část této práce je věnována zabezpečení VoIP komunikace. Pro zabezpečení VoIP komunikace byl vybrán protokol TLS, který je zde podrobně rozebrán. Praktická část je zaměřena na realizaci bezpečného propojení SIP Proxy serverů pomocí protokolu TLS, kde jsou SIP Proxy servery realizovány programy OpenSIPS a Asterisk. Součástí práce je také postup pro vytvoření certifikační autority a následné generace certifikátů.

Klíčová slova

NGN, IMS, SIP, MGCP, Megaco, SIGTRAN, SAML, TLS, SRTP, ZRTP, certifikační autorita, certifikát, OpenSIPS, Asterisk

Abstract

Work is intent on security interdomain connection between SIP Servers. Opening charter treat of architecture and protocols in NGN networks. Sequentially are illustration principles for location destination SIP Proxy server. Best part of this work is devoted security VoIP communication. For security VoIP communication was chosen TLS protocol, which is detail described here. Practical part is bent on realization security connection SIP Proxy servers by means of TLS protocol, where SIP Proxy servers are realized programs OpenSIPS and Asterisk. Part of work is also procedur efor creation certification authorities and resulting generation certificates.

Key words

NGN, IMS, SIP, MGCP, Megaco, SIGTRAN, SAML, TLS, SRTP, ZRTP, certification authority, certificate, OpenSIPS, Asterisk

Seznam použitých symbolů a zkratek

AES	(Advanced Encryption Standard)	Symetrická bloková šifra.
DoS	(Denial of Service)	Internetový typ útoku na základě přehlcení požadavky.
DSS	(Decision Support System)	Systémy na podporu rozhodování.
HP	(Handshake Protocol)	Protokol u TLS pro vyjednání kryptografických informací.
HTTP	(Hypertext Transfer Protocol)	Protokol pro výměnu hypertextových dokumentů.
IMS	(Internet protocol Multimedia Subsystem)	Internetový protokol pro 3G a fixní telefonní sítě.
IP	(Internet Protocol)	Protokol pro přenos dat přes paketové sítě.
ITU-T	(International Telecommunication Union)	Mezinárodní telekomunikační unie.
MAC	(Message Authentication Code)	Autentizační kód zprávy.
MGCP	(Media Gateway Control Protocol)	Protokol ovládající Media Gateway.
MTP	(Media Transfer Protocol)	Část SS7, která se používá pro komunikaci v PSTN.
NGN	(Next Generation Network)	Síť nové generace.
PLMN	(Public Land Mobile Network)	Veřejná pohyblivá pozemní síť.
PSTN	(Public Switched Telephony Network)	Veřejná přepínaná telefonní síť.
QoS	(Quality of Service)	Garance kvality poskytovaných služeb.
RTP	(Real-time Transport Protocol)	Protokol pro přenos zvukových a obrazových dat .
RURI	(Request URI)	Určuje další skok zprávy.

S/MIME	(Secure/ Multipurpose Internet Mail Extensions)	Metoda pro zabezpečení obsahu zprávy.
SAML	(Security Assertion Markup Language)	Standard pro výměnu autorizačních a autentizačních informací.
SCTP	(Stream Control Transmission Protocol)	Protokol pro přenos telefonní signalizace SS7 po IP.
SDP	(Session Description Protocol)	Protokol pro popis vlastností relace.
SGW	(Signalling Gateway)	Brána pro propojení různých signalizačních sítí.
SIP	(Session Initiation Protocol)	Protokol pro přenos signalizace v internetové telefonii.
SRTP	(Secure Real-time Transport Protocol)	Zabezpečený přenos RTP dat.
SS7	(Signalling System Number 7)	Síťová signalizace užívaná v ISDN a GSM sítích.
SSL	(Secure Socket Layer)	Předchůdce protokolu TLS.
TCP	(Transmission Control Protocol)	Internetový protokol pro spolehlivý přenos dat.
TLS	(Transport Layer Security)	Protokol pro zabezpečení různých druhů komunikace.
UA	(User Agent)	Reprezentuje koncový terminál sítě.
UDP	(User Datagram Protocol)	Internetový protokol pro nespolehlivý přenos dat.
URI	(Uniform Resource Identifier)	Jednotný identifikátor zdroje pro přesnou specifikaci zdroje informací.
VoIP	(Voice over Internet Protocol)	Přenos hlasu prostřednictvím IP protokolu.
XML	(Extensible Markup Language)	Značkovací jazyk využívaný standardem SAML.
ZRTP	(Ziemmermann Real-time Transport Protocol)	Rozšíření SRTP o počáteční bezpečnou výměnu klíčů.

Obsah

1 Úvod

Stávající telekomunikační sítě jsou postaveny na dvou různých sítích. Jedná se o klasické telefonní sítě PSTN a o datové sítě založené na paketovém přenosu. Telekomunikační sítě prošly evolucí, ve které můžeme najít řadu milníků. Jedním z nich je IP telefonie (VoIP). Jak klasické PSTN sítě, tak i datové sítě mají své nesporné výhody a nevýhody. Zhruba před 10 lety vznikla koncepce sítě nové generace, která byla založena na myšlence oddělení transportní úrovně od řídící úrovně. Systém NGN obsahuje celou řadu protokolů a jedním z nejpoužívanějších protokolů je SIP protokol. Prvním a zatím jediným standardizovaným systémem NGN je systém IMS, který je založen právě na protokolu SIP

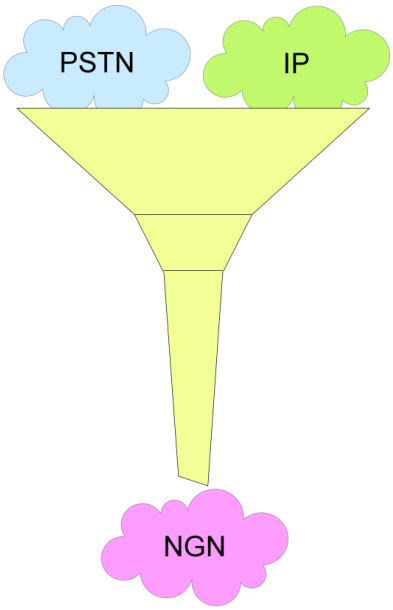
Vraťme se ještě zpátky k výhodám a nevýhodám PSTN a IP sítí. Hlavním nedostatkem klasických PSTN sítí jsou jejich značné finanční náklady při telefonování a to zvláště při telefonování ze zahraničí. Tento neduh odstraňuje VoIP technologie neboli telefonování prostřednictvím internetu. Jelikož PSTN je ve svém principu uzavřená síť, nedochází u ní k rapidním problémům ohledně zabezpečení sítě. U VoIP technologie je tomu právě naopak. VoIP komunikace je založena na bázi otevřeného IP protokolu. Proto je mnohem náchylnější na různé útoky, než klasická PSTN síť. Bezpečnost VoIP komunikace mohou ohrožovat téměř všechny útoky, které se vyskytují v internetu. Proto se v dnešní době klade velký důraz na zabezpečení VoIP. U VoIP komunikace můžeme zabezpečit buď signalizační část, nebo uživatelskou část, nebo obojí.

Praktická část této práce je zaměřená právě na zabezpečení VoIP komunikace a to zvláště její signalizační části. Signalizace je přenášena prostřednictvím SIP protokolu. Pro zabezpečení byl vybrán protokol TLS. Protokolem TLS byla zabezpečena komunikace mezi SIP Proxy servery, jež byly realizovány open-source pobočkovými ústřednami (PBX) OpenSIPS a Asterisk. TLS protokol se neobejde bez certifikátů, a proto je část praktické práce věnována právě tvorbě certifikační autority a generování certifikátů touto autoritou.

2 Architektura a protokoly v NGN

2.1 NGN

Telekomunikační sítě jsou tvořeny dvěma druhy sítí. Jsou jimi telefonní sítě a datové sítě. Jak telefonní, tak i datové sítě mají své výhody a nevýhody. Jednou z velkých výhod klasických telefonních sítí (PSTN) je, že zaručují spolehlivost přenosu informací, což se o datových sítích říct nedá. Naproti tomu datové sítě vynikají efektivnějším využitím poskytnuté přenosové cesty. Postupem času se začaly telefonní a datové sítě více či méně prolínat.
Naprostým sloučením telefonních a datových sítí a využitím výhod obou těchto metod vznikla síť nové generace neboli NGN (Next Generation Network).



Obr.1: Sloučení telefonních a datových sítí za vzniku NGN

Znaky a vlastnosti NGN sítí si mnoho lidí vysvětluje různě. Institut ITU-T vydal následující definici NGN:
"A Next Generation Network (NGN) is a packet-based network able to provide services including Telecommunications Services and able to make use of multiple broadband, QoS-enabled transport technologies and in which service-related functions are independent from underlying transport-related technologies.

It offers unrestricted access by users to different service providers. It supports generalized mobility which will allow consistent and ubiquitous provision of services to users."

"NGN je paketová síť poskytující telekomunikační služby, která umožňuje využívat různé širokopásmové transportní technologii zaručující kvalitu poskytovaných služeb (QoS), kde jednotlivé služby jsou nezávislé na použité transportní technologii [7], [19].
Dále poskytuje uživatelům neomezený přístup k různým poskytovatelům služeb. Podporuje mobilitu uživatelů a umožňuje jim všeobecné a nepřetržité využívání poskytovaných služeb."

Z definice vyplývá jedna velmi důležitá vlastnost, a to oddělení transportní úrovně nesoucí uživatelská data od řídicí úrovně, která zajišťuje sestavování, dohled a údržbu nad spojením. Taktéž řídicí úroveň je oddělena od aplikační úrovně, jež obsahuje služby, které jsou poskytovány jednotlivými providery.
NGN obsahuje prvky zvané brány (gateway), které zajišťují konverzi mezi různými typy dat. NGN obsahuje dva typy bran:

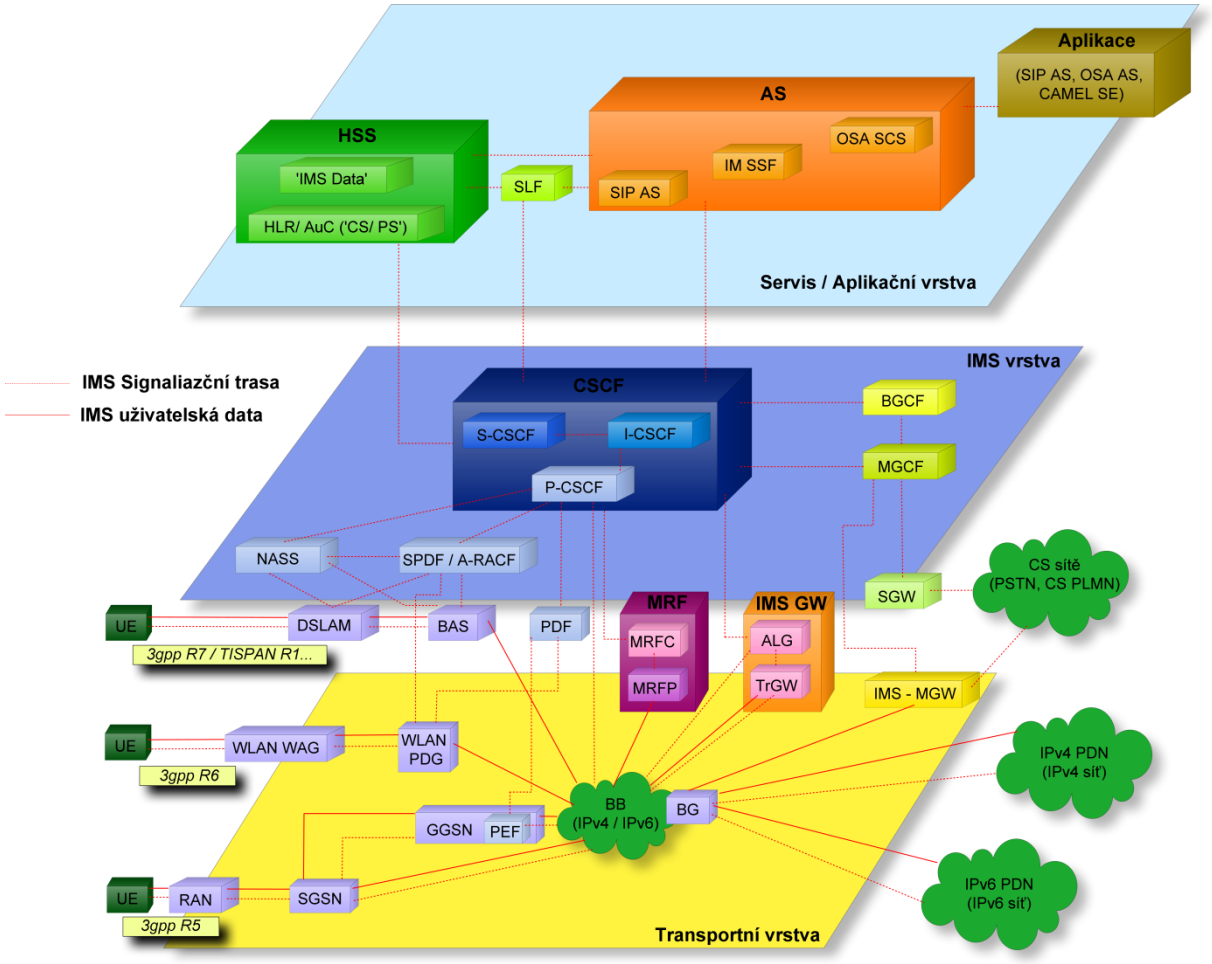
- MGW (Media Gateway) – slouží pro převod uživatelských informací.
- SGW (Signalling Gateway) – slouží pro převod řídicích informací.

Řídicím prvkem NGN sítí je MGC (Media Gateway Controller), který obsluhuje jednotlivé brány.
Prvním standardizovaným systémem, který byl uveden na trh, je systém IMS.

2.2 IMS

IMS (Internet protocol Multimedia Subsystem) je projekt organizace 3GPP (3rd Generation Partnership Project), jejímž úkolem je specifikace mobilních systémů třetí generace (systémem druhé generace je GSM) [7], [12]. Organizace 3GPP vydalo několik specifikací, ale až v roce 2000 vydalo „Vydání 5", ve kterém představilo koncept IMS. IMS je založen na IP principu, který spolupracuje se stávajícími datovými a hlasovými sítěmi. Úkolem IMS je nahradit dosavadní rozdělení na sítě s přepínáním okruhů a sítě na základě IP jediným standardem s jediným typem přístupu, a to paketovým přístupem (All – IP). IMS architektura umožňuje navázat spojení typu bod – bod. Taktéž umožňuje zaručit kvalitu poskytovaných služeb QoS a poskytuje služby v reálném čase. IMS využívá celou řadu protokolů definovaných v IETF(Internet Engineering Task Force), ale nejvíce využívá protokolu SIP.

Architektura IMS



Obr.2: Schéma architektury IMS

Architektura IMS v sobě zahrnuje řadu nových funkčních bloků. Hlavními prvky IMS architektury jsou:

- CSCF (Call Session Control Function) neboli funkce ovládání hovoru, jež mají na starosti směrování, alokaci zdrojů a řízení vlastností spojení. Pro signalizaci využívá SIP protokol, pro užitečné informace RTP protokol a pro komunikaci s databázemi protokol Diameter. Existují tři druhy CSCF:
    - P-CSCF(Proxy Call Session Control Function) představuje první kontaktní bod pro uživatele. Je to v podstatě Proxy server, který přesměrovává žádosti a odpovědi a autentizuje uživatele. V IMS se uživatel autentizuje na P-CSCF a na ostatních entitách se již nemusí autentizovat, neboť ostatní entity věří P-CSCF. Další vlastností je

komprese a dekomprese SIP zpráv a generování účtovacích informací. Tento prvek je v IMS síti povinný.

- I-CSCF (Interrogating Call Session Control Function) představuje SIP Proxy umístěnou uvnitř administrativní domény. Základní funkcí je nalezení HSS serveru uživatele a na základně informací z HSS směrovat SIP žádosti a odpovědi do konkrétního S-CSCF.

- S-CSCF (Serving Call Session Control Function) je v podstatě mozkem IMS sítě. Dohlíží nad spojením, zajišťuje registraci a na základě uživatelského profilu směruje SIP zprávy na příslušné aplikační servery.

- HSS (Home Subscriber Server) je hlavní databází, která obsahuje informace o účastnících a službách.

- AS (Application Servers) neboli aplikační servery poskytují aplikační služby. Jednotlivé IMS domény mohou podporovat více aplikačních serverů pro různé služby.

- MGCF (Media Gateway Control Function) zajišťuje spolupráci mezi telefonní a IP sítí.

- BGCF (Breakout Gateway Control Function) je v podstatě SIP server, který provádí směrování na základě telefonního čísla. BGCF je využíván v relacích, které jsou inicializovány od IMS terminálu a adresovaný uživatel je v síti s přepojováním okruhů (např. PSTN nebo PLMN).

- SGW (Signalling Gateway) zajišťuje propojení různých vnitřních signalizačních sítí.

- MRFC (Media Resource Function Controler) zpracovává požadavky od S-CSCF a aplikačního serveru (AS) a ovládá tok dat z MRFP.

- MRFP (Media Resource Function Processor) je ovládán pomocí MRFC a podporuje řadu funkcí (např. generování tonů a hlášek).

- SLF (Subscription Locator Register) je databáze pro vyhledávání adresy HSS, který náleží konkrétnímu uživateli.

- RAN (Radio Access Network) je rádiová přístupová síť, která poskytuje rádiové rozhraní.

- SGSN (Serving GPRS Support Node) spojuje rádiovou a paketovou síť.

- GGSN (Gateway GPRS Support Node) zprostředkovává propojení s vnějšími paketovými sítěmi.
- UE (User Equipment) terminál uživatele.

2.3 SIP

2.3.1 Popis SIP protokolu

Protokol SIP (Session Initiation Protocol) je řídicí protokol, který pracuje na aplikační vrstvě. SIP protokol řídí sestavování, modifikaci a ukončení spojení s jedním či více účastníky. SIP protokol byl vyvíjen od roku 1996 skupinou MMUSIC [1]. Během standardizace SIP protokolu bylo vydáno několik standardů RFC, ale až ve standardu RFC 3261 bylo popsáno hlavní jádro SIP protokolu, jak ho známe dnes.
Pro VoIP komunikaci nám nepostačí jenom SIP protokol, ale musíme využít ještě dalších dvou protokolů. Těmito protokoly jsou RTP (Real-time Transport Protocol) protokol pro přenos užitečné informace v reálném čase a SDP (Session Description Protocol) protokol pro vyjednání parametrů spojení.

SIP je end-to-end orientovaný signalizační protokol. Jelikož SIP je typu end-to-end, tak z toho plyne, že veškerá logika je uložena v koncových zařízeních. Další charakteristikou vlastností SIP protokolu je jeho textová orientace. Textově orientovaný SIP protokol má velmi mnoho podobných rysů a vlastností s protokolem HTTP. Textová orientace SIP protokolu je jeho velmi velkou výhodou. Hlavně v začátcích jeho fungování, kdy jeho struktura byla velmi jednoduchá a přehledná. Nyní se považuje SIP za velmi složitý protokol.

Specifickým prvkem SIP protokolu je SIP doména. SIP entita je vázána právě k doméně, která je obsluhovaná určitou SIP Proxy neboli SIP serverem. SIP entity jsou identifikovány pomocí SIP URI (Uniform Resource Identifier). Tvar SIP URI vypadá následovně.

sip:user:password@host:port,uri-parametrs?headers

Mnohem častěji se setkáme s jednodušší podobou SIP URI, která se skládá pouze z uživatelské části (user) a hostitelské části (host) respective doménové části, kde obě části jsou od sebe odděleny znakem „@“. Tato jednodušší forma SIP URI vypadá následovně.

sip:user@host neboli např. *sip:1111@home.cz* nebo *sip:728963245@158.196.244.210*

2.3.2 Základní SIP prvky

Základními SIP prvky jsou:

- **Uživatelští agenti (UA)** – reprezentují koncové terminály sítě, kterými mohou být hardwarové nebo softwarové telefony.
  - UAC (User Agent Client) – vysílá požadavky a přijímá odpovědi
  - UAS (User Agent Server) – přijímá požadavky a vysílá odpovědi
  - B2BUA (Back-to-Back User Agent) – speciální UA, který je vkládán do cesty za účelem vytváření dvou spojení. Na B2BUA jedno spojení končí a nové spojení je sestaveno dále k cíli. Z pohledu koncových zařízení se chová jako SIP server.

  Každé koncové zařízení obsahuje v sobě jak UAC, tak i UAS

- **SIP servery**
  - Proxy
    - Stateless
    - Stateful
  - Redirect
  - Registrar
  - Location

SIP Proxy servery zajišťují směrování žádosti o spojení (INVITE) podle aktuálního umístění volaného. Kromě této základní vlastnosti může SIP Proxy zajišťovat autentizaci, účtování, různé doplňkové služby, atd. Podrobný popis SIP serverů nalezneme v příloze (Příloha X.).

2.3.3 SIP komunikace

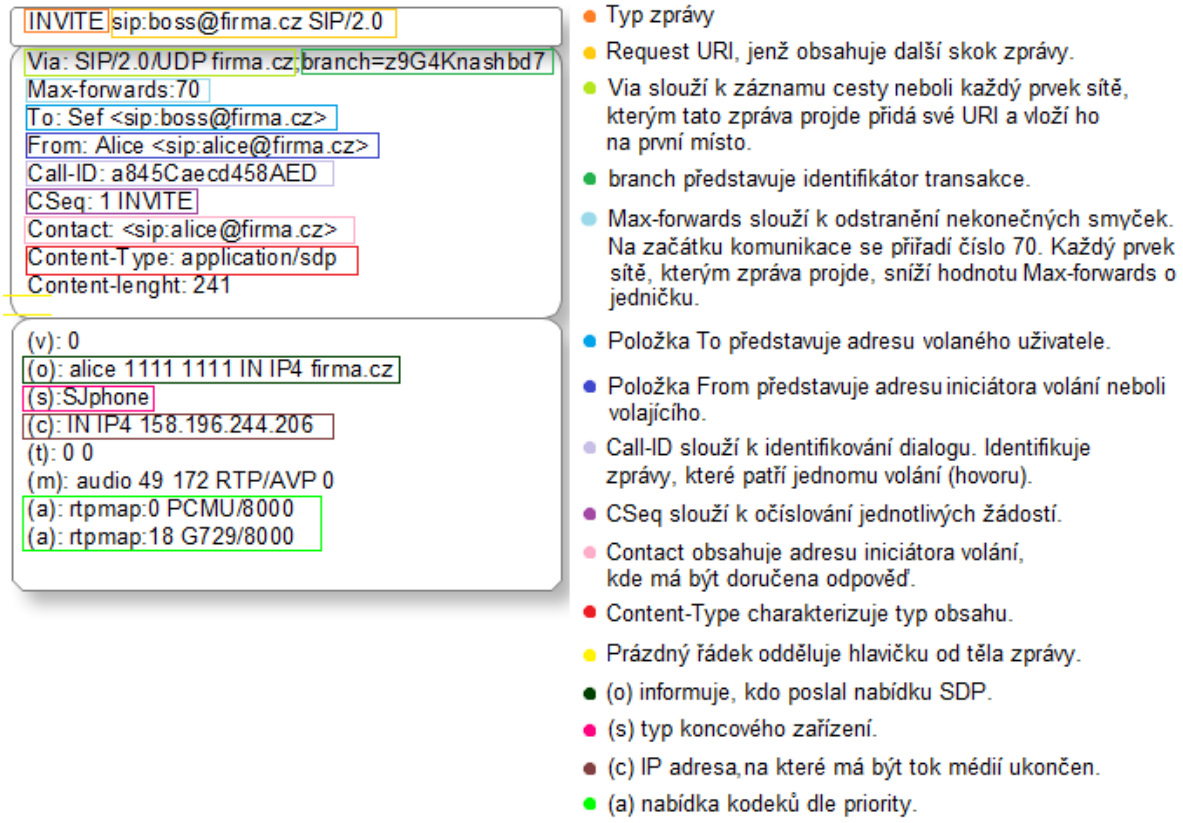
Žádosti a odpovědi

SIP komunikace používá dva základní typy zpráv. Prvním typem je žádost a druhým odpověď.

Žádosti jsou používány k sestavení, aktualizaci nebo ukončení spojení. Ve standardu RFC 3261, který popisuje hlavní jádro SIP protokolu, je popsáno šest základních typů žádostí (INVITE, ACK, BYE, CANCLE, REGISTER a OPTIONS). Kromě těchto základních žádostí, bylo v pozdějších RFC popsáno ještě 8 dalších typů žádostí (INFO, PRACK, SUBSCRIBER, NOTIFY, UPDATE, MESSAGE, REFER a PUBLISH).

Každá SIP komunikace začíná vždy zprávu typu „žádost“. Po žádosti následuje vždy odpověď. Jako všude, tak i v SIP komunikaci existuje nějaká výjimka. Každá žádost je potvrzena odpovědí, kromě žádosti ACK. Žádost ACK je žádostí a zároveň i odpovědí. Je to žádost, která potvrzuje doručení odpovědi na zprávu INVITE. Při SIP komunikaci se můžeme setkat z šesti druhy odpovědí. Jsou jimi odpovědi 1xx až 6xx.

Popis SIP hlavičky



Obr.3: Popis SIP hlavičky

Transakce

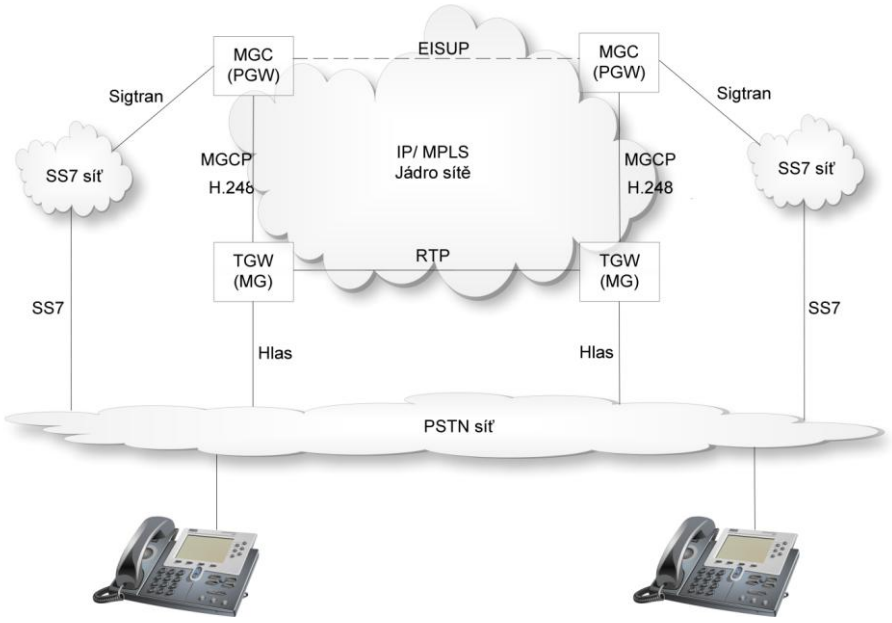
Transakce jsou u SIP komunikace z pohledu toku zpráv a zacházení s těmito zprávami velmi důležité a to z toho důvodu, že většina prvků při SIP komunikaci jsou transakční zařízení. Transakcí rozumíme skupinu SIP zpráv, které jsou vyměňovány mezi SIP síťovými prvky. Transakce obsahuje jednu žádost a všechny odpovědi na ni. Identifikátor umožňující porovnání transakcí se nazývá *branch*. Tento identifikátor *branch* se nachází v hlavičce SIP zprávy v poli *Via*.

Dialog

Dialog je soubor SIP zpráv mezi dvěma komunikujícími body. Dialogy popisují souslednost a směrování SIP zpráv mezi koncovými terminály. Dialogy jsou identifikovány pomocí tří položek. Jsou jimi pole z SIP hlavičky s názvem *From (tag)*, *To (tag)* a *Call-ID*. Zprávy, které mají tyto tři pole z SIP hlavičky stejné, patří jednomu dialogu. Dialog tvoří jak žádosti, tak odpovědi. Ovšem ne všechny odpovědi typu 1xx a 6xx jsou schopny tvořit dialog. Pouze odpovědi typu 101 až 199 jsou schopny tvořit dialog, jelikož pouze tyto odpovědi obsahují pole *To (tag)*.

2.4 MGCP a Megaco

MGCP (Media Gateway Control Protocol) protokol byl vydán v roce 1999 ve specifikaci RFC 2705, standardizace se však dočkal až o 4 roky později v RFC 3435 [14]. MGCP protokol balí své pakety do UDP protokolu. Protokol Megaco/H248 vychází z protokolu MGCP. Megaco je výsledkem spolupráce skupiny IETF (Engineering Task Force) a ITU-T. Termín Megaco je označení skupinou IETF. ITU později převzala vlastnictví protokolu od IETF a dala protokolu nový název, a to H.248.



Obr.4: Schéma MGCP sítě

MGCP je protokol typu Master/Slave, kdy Master je Call Agent, který ovládá Media Gateway (Slave). Hlavní funkce protokolu tkví v umožnění řízení telefonních bran (Media Gateway) z externích objektů, jako jsou MGC (Media Gateway Controller) a CA (Call Agent). Tyto externí objekty umožňují posílání a přijímání příkazů do a z brány a také řízení a kontrolu signálů. Brány zabezpečují konverzi mezi sítí s přepojováním okruhů a paketovou sítí. MGCP protokol spolupracuje jak s protokolem SIP, tak i s H.323 protokolem. V sítích, kde se využívá spolupráce protokolu MGCP a protokolů SIP a H.323 má následující rozdělení funkcí. MGCP protokol se stará o řízení bran a protokoly SIP a H.323 o řízení vlastního hovoru.

2.4.1 Hlavní prvky MGCP architektury

- **Media Gateway (MG)**

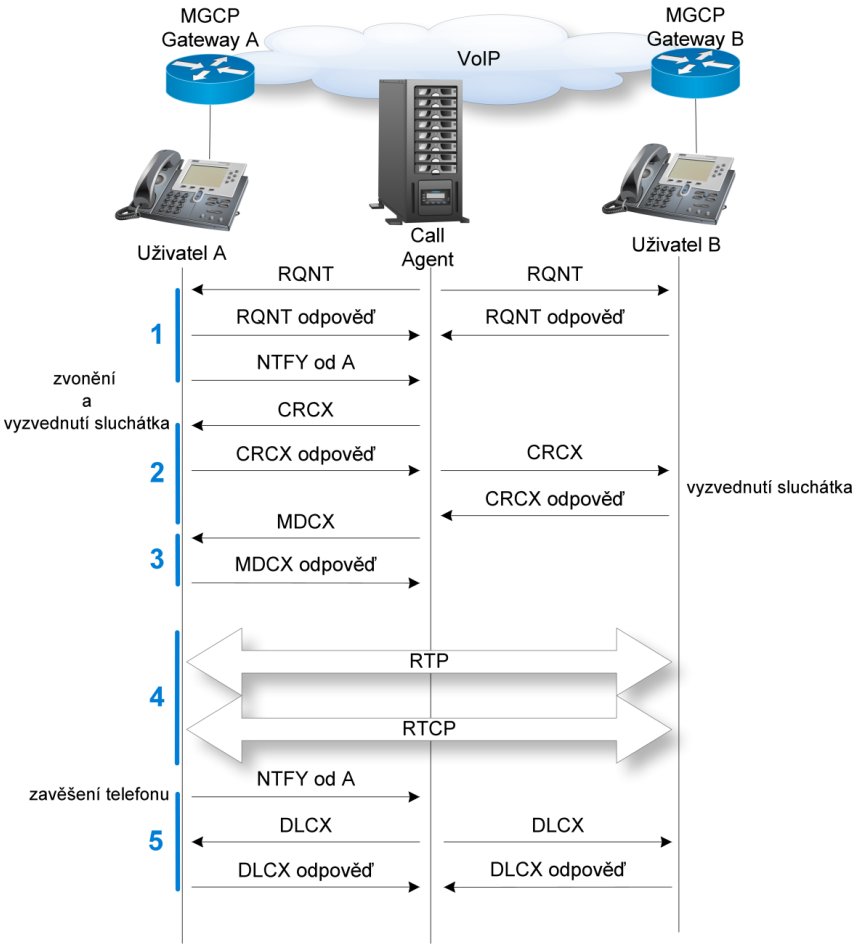
  Media Gateway je prvek, který obstarává funkci konverze dat mezi daty přenášenými klasickou telefonní sítí a daty přenášenými pomocí IP protokolu.

- **Media Gateway Controller (MGC) neboli Call Agent (CA)**

  Call Agent řídí Media Gateway prostřednictvím signalizačního protokolu MGCP (Media Gateway Controller Protocol). Call Agent skrze MG obstarává řídicí funkce, jako jsou sestavení, modifikace, kontrola a rozpad spojení s multimediálními koncovými body (uživateli). Obrázek níže zobrazuje klasické využití protokolu MGCP.

Pakety MGCP jsou dvojího typu, buď se jedná o žádost, nebo o odpověď (stejně jako u protokolu SIP). Za každou odeslanou žádostí následuje patřičná odpověď. Podrobný popis žádostí a odpovědí je přiložen v příloze (Příloha XI.).

Průběh MGCP relace



Obr.5: Průběh MGCP relace

1. Call Agent (CA) žádá Media Gateway (MG), aby mu poskytla notifikaci o sobě a jejich uživatelích. MG tak neprodleně učiní. Zpráva RQNT často obsahuje důležité informace (např. signalizační pakety a Dial mapu), takže MG může shromažďovat vytáčené číslice od uživatele ještě předtím, než pošle zprávu NTFY ke Call Agentovi
2. Call Agent říká zprávou (CRCX) MG „A“, aby vytvořila požadované spojení a definovala port pro RTP média. MG odpoví zprávou (CRCX odpověď). CA pošle zprávu CRCX i MG „B“, která taktéž odpoví.
3. Call Agent nyní říká MG „A“, aby upravila své parametry relace a porovnala je s MG „B“.
4. Začátek RTP medií.
5. Ukončení hovoru. Uživatel „A“ zavěsí sluchátko a MG „A“ notifikuje tento krok Call Agentovi. Call Agent poté informuje MG „A“ o ukončení hovoru (DLCX). MG „A“ potvrdí ukončení hovoru. Call Agent informuje o rozpadu spojení i MG „B“. Taktéž MG „B“ potvrdí ukončení spojení.

2.5 SIGTRAN

Pracovní skupina IETF (Internet Engineering Task Force) navrhla novou sadu protokolů pro přenos SS7 signalizačních zpráv přes IP síť nazvanou SIGTRAN. Soubor protokolů se skládá z nového transportního protokolu a různých upravených protokolů. Používání SIGTRAN protokolů je prvním krokem ke sloučení SS7 sítí s IP sítěmi. SIGTRAN je schopen vyřešit spojení s izolovanými ostrůvky SS7 sítí, které by jinak vyžadovaly drahou SS7 infrastrukturu. V dnešní době se většina telekomunikačních společností ubírá směrem k IP sítím, které budou postupně nahrazovat tradiční telefonní sítě [13], [17].

Přes internet jsou zprávy přenášeny pomocí TCP a UDP transportního protokolu. Tyto transportní protokoly ovšem nesplňují všechny podmínky pro přenos real-time signalizace. Těmito podmínkami jsou:

- Souslednost, spolehlivost přenosu
- Redundance z důvodu výpadku linky
- Nízká ztrátovost a zpoždění
- Ochrana proti DoS

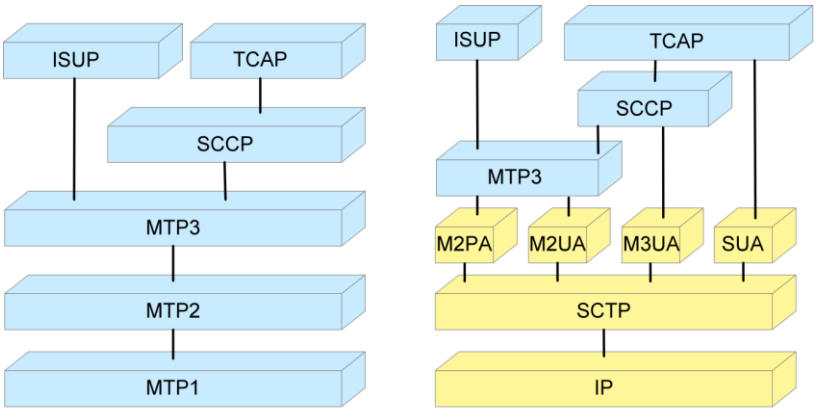
Z tohoto důvodu byl navržen nový transportní protokol zvaný SCTP (Stream Control Transmission Protocol).

2.5.1 SIGTRAN architektura

SIGRAN obsahuje transportní protokol SCTP a několik uživatelsky adaptačních protokolů, které jsou nezbytné pro přenos SS7 zpráv přes IP. SIGTRAN architektura obsahuje tři vrstvy:

- IP vrstva
- Transportní vrstva (SCTP)
- Uživatelsky adaptační vrstva (M2PA, M2UA, M3UA a SUA)

  Vlastnosti uživatelské vrstvy jsou podrobně popsány v příloze (Příloha XII.).



Obr.6: Schéma SIGTRAN architektury

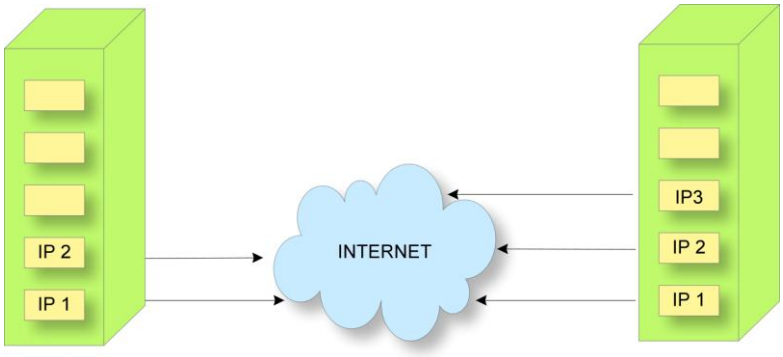
Obrázek (*Obr.6*) zobrazuje tři nižší vrstvy protokolu SIGTRAN. Tyto vrstvy nahrazují spodní vrstvy SS7 protokolu (MTP1 a MTP2), které umožňují přenos přes IP síť. SCTP je transportní protokol stejně jako TCP, ovšem s několika vylepšeními.

2.5.2 SCTP protokol

Jak už jsem zmínil výše, SCTP protokol je jediným transportním protokolem, který vyhovuje přísným požadavkům pro přenos SS7 signalizačních zpráv, jako jsou velmi nízká ztrátovost a zpoždění. SCTP protokol je podobný TCP, ale má řadu odlišnosti. Hlavními odlišnostmi jsou multi-homing a multi-streaming.

Multi-homing

Multi-homing je jeden uzel s několika IP adresami, kde každá dvojice IP adres mezi dvěma uzly se nazývá cesta (path). V SCTP spojení si může každý uzel vybrat pouze jednu primární cestu a přes ni jsou posílána data. Pokud je primární cesta nedostupná, přejde uživatel na jinou cestou, která je dostupná a tou poté pošle data.



Obr.7: Multi-homing

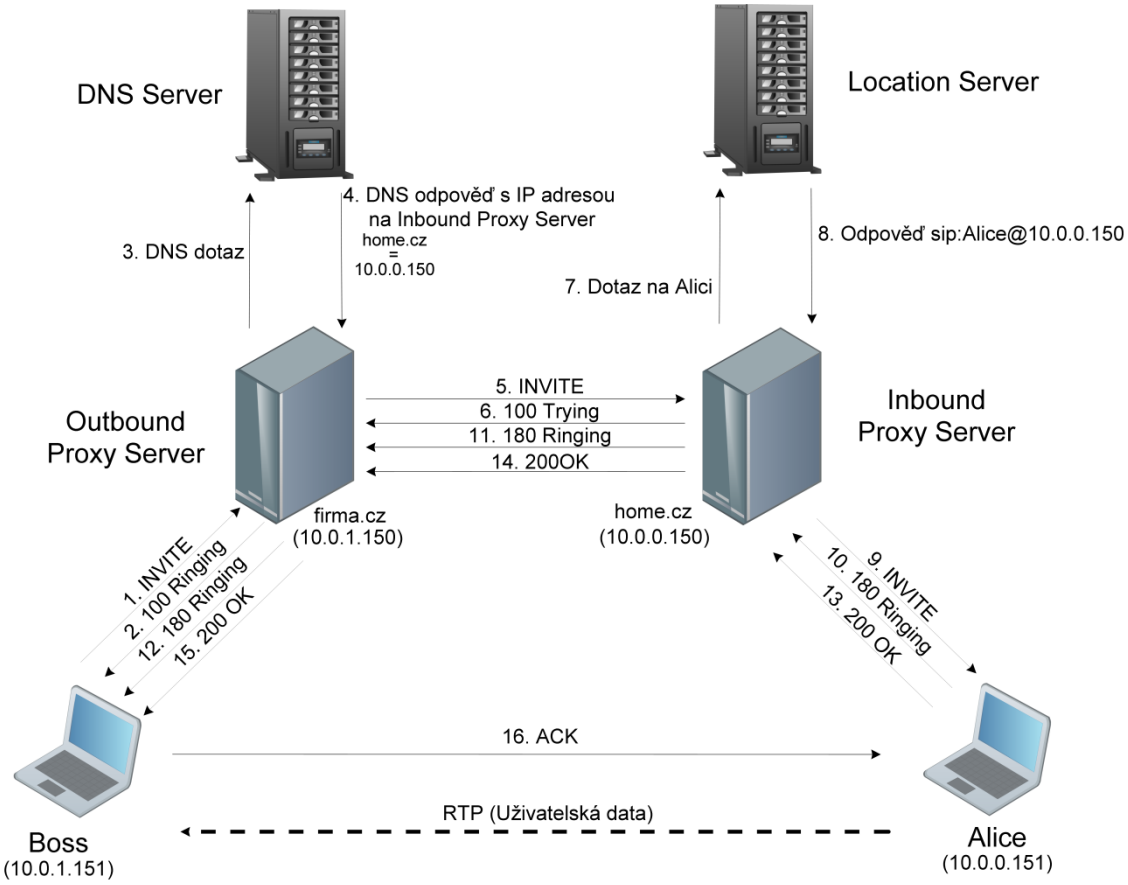
Multi-streaming

Multi-streaming zabraňuje ztrátě bloků data při přenos. V SCTP asociaci může mezi dvěma uzly probíhat několik streamů, kde každý je přiřazený konkrétnímu zdroji nebo aplikaci. Tyto streamy se vzájemně neblokují a tím pádem dochází k velkým ztrátám paketů a ke zpoždění.

3 Scénáře nalezení cílového SIP Proxy serveru

Každá firma, která využívá VoIP komunikaci, vlastní zpravidla jeden SIP Proxy server. Tento SIP Proxy spravuje všechny uživatelské agenty (UA neboli terminály) v rámci této sítě. SIP Proxy reprezentuje většinou jednu doménu (např. firma.cz) [1]. Nemusí tomu být takhle vždycky, SIP Proxy může obsluhovat i více domén a potom se jedná o multidoménovou SIP Proxy. Multidoménová SIP Proxy určuje, který UA patří do které domény pomocí pole *realm*. Ve většině případů SIP Proxy obsluhuje pouze jednu doménu. Na následujícím příkladu bude vysvětlen princip nalezení cílového SIP Proxy (home.cz) zdrojovým SIP proxy (firma.cz).

UA šéfa firmy (*boss@firma.cz*) chce volat Alici domů (*alice@home.cz*). Šéf použije adresu *sip:alice@home.cz*. UA šéfa sice neví, kam má poslat žádost na sestavení spojení, ale je naprogramován tak, aby veškerý odchozí provoz směroval na SIP Proxy server své firmy (*firma.cz*). Proxy server si už s tímto požadavkem dokáže poradit, a to dvěma způsoby. Buď pomocí statického mapovaní, nebo pomocí DNS serveru. Statické mapování znamená, že Proxy server (*firma.cz*) má ve své směrovací tabulce staticky nakonfigurovaný záznam, že doména *home.cz* se nachází na IP adrese 10.0.0.150. Tím pádem ví, že žádost o sestavení spojení má poslat na adresu 10.0.0.150, tedy na doménu *home.cz*. Proxy server přepošle *INVITE* od *boss@firma.cz* přímo na Proxy server (*home.cz*). Proxy server (*home.cz*) lokalizuje Alici pomocí Location serveru a přepošle *INVITE* na UA Alice (*alice@home.cz*). Druhý způsob je ten, že SIP Proxy nemá ve směrovací tabulce žádný záznam o tom, kde se nachází doména *home.cz*. Tím pádem Proxy server (*firma.cz*) vyhledá záznam o doméně pomocí SRV záznamu v DNS serveru. Z SRV záznamu se dozví, že doména *home.cz* se nachází na IP adrese 10.0.0.150. Proxy server (*home.cz*) poté přepošle přijatý *INVITE* přímo na Proxy server (*home.cz*). Proxy server (*home.cz*) lokalizuje Alici pomocí Location serveru a přepošle *INVITE* na UA Alice (*alice@home.cz*). Poté následují potvrzující zprávy vyzvánění telefonu Alice (*100 Ringing*) a potvrzení vyzvednutí sluchátka Alicí (*200 OK*). Šéf potvrdí začátek hovoru zprávou (*ACK*). A následuje vlastní hovor přenášený pomocí RTP protokolu.



Obr.8: Schéma komunikace při nalezení cílového SIP Proxy serveru

4 Způsoby zabezpečení

4.1 SAML

SAML (Security Assertion Markup Language) byl vyvinut technickou komisí bezpečných služeb OASIS [9], [16]. Cílem OASIS je vyvinutí standardu pro výměnu autorizačních a autentizačních informací. SAML je systém založený na jazyku XML, jehož úkolem je výměna autorizačních a autentizačních informací mezi entitami, neboli jinými organizacemi, aplikacemi, atd. Pro snazší pochopení si vysvětlíme SAML na následujícím příkladu: Uživatel „A“ se autentizuje na portálu (first). Portál (first) nejenom, že autentizuje uživatele „A“, ale ještě ví o uživateli „A“, že má nějakou funkci (roli). Tuto informaci přidá do tvrzení v SOAP (Service Oriented Architecture Protocol, neboli protokol servisně orientované architektury) zprávě. Tato SOAP zpráva je poslána na další portál (second). Portál (second) se při obdržení SOAP zprávy podívá na portálovou identitu, ověří digitální podpis portálu „first“ a povolí či zakáže přístup uživatele „A“ vzhledem k jeho funkci (roli).

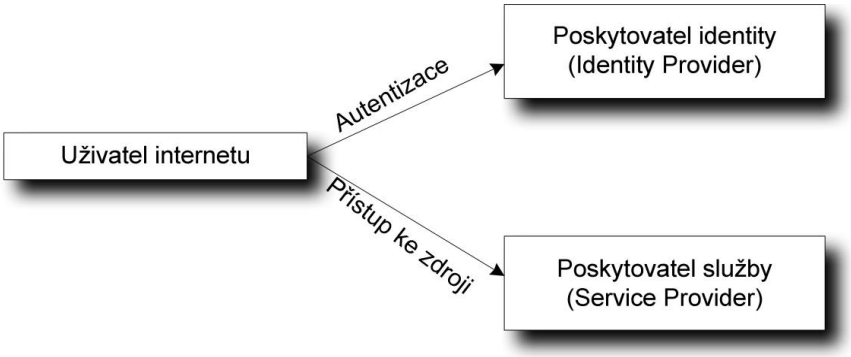
SAML plní tři základní úkoly:

- Definice sémantiky a syntaxie XML zpráv obsahující tvrzení (assertion) v XML formě.
- Definice protokolů žádostí a odpovědí (request-responce) mezi žádající a odpovídající stranou při výměně bezpečnostních informací.
- Definice pravidel pro používání tvrzení (assertion) s transportními standardy (např. transport SAML assertion ve zprávě SOAP prostřednictvím HTTP).

Základní využívání SAMLu:

- Jednotné přihlášení na internetu (Web Single Sign-On neboli Web SSO)

  Uživatel se přihlásí na webové stránce (*auto.cz*) a je poté autentizován. Pokud se chce později přihlásit na jinou webovou stránku (*motorky.cz*), musel by bez využití Web SSO opět zadávat své přihlašovací údaje a autentizovat se. Pokud uživatel využil při prvním přihlášení SAML, nemusí se na webové stránce (*motorky.cz*) opět přihlašovat a autentizovat. Webová stránka (*motorky.cz*) vyšle požadavek na (*auto.cz*), zda se u ní uživatel autentizoval. *Auto.cz* pošle odpověď, že ano. Tím pádem webová stránka (*motorky.cz*) již nepožaduje po uživateli přihlašovací údaje a rovnou mu poskytne své zdroje.



Obr.9: Příklad využití Web SSO

- Autorizace založená na atributech (Attribute-based Authorization)

  Tato metoda je podobná metodě Web SSO. Jedna webová stránka předává informace o uživateli další webové stránce. Informace o uživateli mohou být např. jak a kdy byl daný uživatel autentizován.

- Zabezpečení webových služeb

  SAML může být využit ve zprávách SOAP za účelem přenosu zabezpečených informací a informací o identitě mezi komunikujícími stranami.

SAML obsahuje čtyři komponenty:

1. Tvrzení (Assertions)
2. Protokol
3. Vazba
4. Profil

1. **Tvrzení** je souhrn informací, jenž obsahuje jedno nebo více sdělení vytvořených SAML autoritou. SAML autorita definuje tři druhy tvrzení:

   - ***Autentizace***

     Uvedený subjekt *S* byl autentizován konkrétními prostředky *M* v konkrétním čase *T*. Jinými slovy: Subjekt *Petr* v bezpečnostní doméně *firma.cz* byl autentizován v čase *9:50*.

   - ***Atribut***

     Uvedený subjekt *S* je svázán s atributy *A*, *B* a tím pádem s hodnotami *a*, *b*.

   - ***Rozhodnutí o autorizaci***

     Výsledek rozhodnutí o žádosti povolení přístupu typu *A* subjektu *S* k uvedeným zdrojům *R*. O výsledku žádosti rozhoduje vydávající autorita na základě přítomnosti evidence *E* od subjektu *S*.

2. **Protokoly**

   SAML definuje řadu protokolů, které umožňují poskytovateli služby provádět řadu operací (např. požadavek na poskytovatele služby, aby autentizoval uživatele a vrátil příslušné tvrzení).

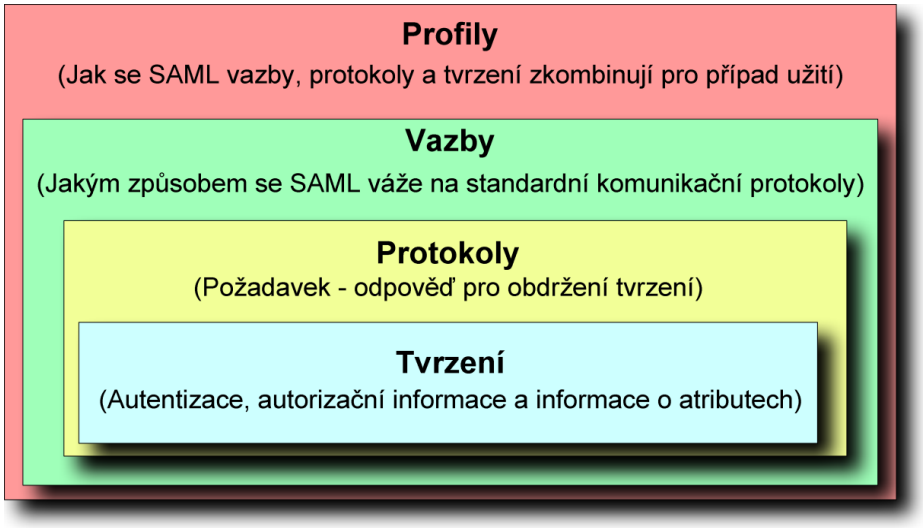
3. **Vazby**

   Definice různých vazeb (např. jak by měly projít SAML zprávy přes HTTP)

4. **Profily**

   Různá rozšíření a omezení SAML pro konkrétní aplikace (např. Web SSO profil).

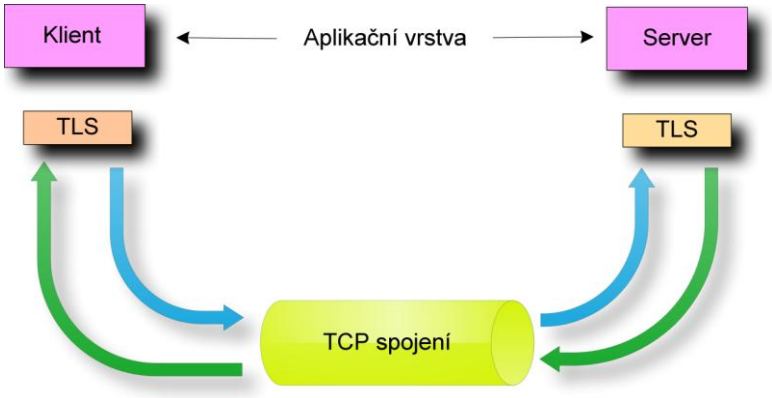
Následující obrázek ilustruje vztah mezi výše uvedenými komponenty.



Obr.10: Vztahy mezi jednotlivými komponenty standardu SAML

4.2 Protokol TLS

TLS protokol slouží k zabezpečení různých druhů komunikace. Protokol TLS zabezpečuje hlavně komunikaci na aplikační vrstvě, ale využívá se i k zabezpečení komunikace na nižších vrstvách. Předchůdcem TLS (Transport Layer Security) protokolu byl protokol SSL (Secure Socket Layer) [5]. TLS využívá jak protokolu TCP, tak i UDP. Využití protokolu TCP je mnohem častější, než UDP. TCP protokol je vytvořen jako duplexní komunikační protokol, což znamená, že jsou vytvořeny dva samostatné kanály. Jeden kanál od klienta k serveru a druhý od serveru ke klientovi.



Obr.11: Vytvoření duplexní komunikace u TCP protokolu

TLS vrstva nezkoumá data poskytnutá aplikační vrstvou, ale pouze přijímá pakety od aplikační vrstvy, které následně zabezpečí. TLS umožňuje zabezpečit pakety pomocí šifrování a integrity dat na základě kryptografického kontrolního součtu MAC (Message Authentication Code). Integrita je zajištěna pomocí MAC počítaného ze sdíleného tajemství a přenášených dat. TLS protokol neobstarává elektronický podpis jednotlivých aplikačních dat v pravém slova smyslu. Ale pro autentizaci serveru a klienta se využívá algoritmus, který se označuje jako elektronický podpis. Je to pouhé označení algoritmu. Elektronický podpis jako název algoritmu využívá dva algoritmy, a to algoritmus RSA a DSS algoritmus.

Protokol TLS tvoří dva základní protokoly.

Prvním je protokol RLP (Record Layer Protocol), jenž přijímá data od aplikační vrstvy a šifruje tyto data a počítá z nich MAC.

Druhým z nich je Handshake protokol. Handshake protokol (HP) je aplikován ihned po navázání spojení mezi klientem a serverem. HP slouží pro vyjednávání kryptografických informací mezi serverem a klientem. Tyto dohodnuté kryptografické informace připraví pro pozdější využití RLP protokolem. HP protokol obsahuje další dva pomocné protokoly.

Protokol CCSP (Change Cipher Specification Protocol), jenž připravuje nachystané kryptografické informace od HP protokolu pro využití RLP protokolem. Vyjednané a dohodnuté kryptografické informace od HP protokolu zkopíruje do tzv. aktuální protokolové svity. Tuto aktuální protokolovou svitu využívá RLP protokol pro šifrování aplikačních dat.

Při jakýchkoliv problémech v komunikaci se aplikuje AP (Alert Protocol) protokol, který je druhým pomocným protokolem.

V následujícím textu bude popsán proces výměny TLS zpráv tzv. TLS Handshake protokol.

4.2.1 Problematika Handshake protokolu

Handshake protokol se z pohledu protokolu RLP (Record Layer Protocol) tváří jako další aplikační protokol a slouží k dohodnutí podmínek vzájemné komunikace mezi klientem a serverem. Tím se má namysli domluvení kryptografických algoritmů a kryptografického materiálu.

Proces vytváření Handshake relace:

1. Zřizováním relace začíná vždy klient a to zprávou *Client Hello* Touto zprávou nabízí klient serveru své podporované kryptografické algoritmy. Tato zpráva obsahuje verzi protokolu TLS; 32B dlouhé náhodné číslo (*client_random*) generované klientem; identifikátor relace (*sessionID*); seznam protokolových svit (*cipherSuite*), kde klient popisuje všechny jím podporované svity (asymetrický protokol, symetrický protokol a protokol pro výpočet otisku) a nakonec seznam podporovaných kompresních algoritmů (*compressionMethod*).

2. Server odpoví na přijatou zprávu řadou zpráv:

   - zprávou *Server Hello*. Touto zprávou server sděluje klientovi, který algoritmus si z nabídky klienta vybral. Tato zpráva je podobná zprávě *Client Hello*. Obsahuje verzi protokolu; náhodné číslo generované serverem (*server_random*); identifikátor relace (*sessionID*); zvolenou protokolovou svitu (*cypherSuite*) a zvolený kompresní algoritmus (*compressionMethod*).

   - Pokud se chce server autentizovat, pošle klientovi ve zprávě *Certificate* svůj certifikát. Tato zpráva obsahuje řetězec certifikátů.

   - Pokud server nezašle svůj certifikát, nebo se zaslaný certifikát nehodí pro zabezpečení přenosu předběžného sdíleného tajemství, odešle klientovi zprávu *ServerKeyExchange*, která obsahuje veřejnou část z vygenerovaných párových dat serverem.

   - Pokud server požaduje autentizaci klienta, pak posílá zprávu *CertificateRequest*, jenž obsahuje seznam jedinečných jmen důvěryhodných certifikačních autorit klientů a seznam typů podporovaných certifikátů.

   - Nakonec server odešle prázdnou zprávu *SeverHelloDone*, jenž klientovi signalizuje, že server skončil a je řada na něm.

3. Klient odpoví řadou zpráv:

    - Pokud klient nezamýšlí k serveru přistupovat anonymně, pošle ve zprávě *Certificate* svůj certifikát.

    - Dále následuje důležitá zpráva *ClientKeyExchange*, jenž obsahuje předběžné tajemství šifrované veřejným klíčem z certifikátu serveru. [5] Z tohoto tajemství se vypočítá hlavní tajemství a tím pádem provede autentizace serveru.

    - Ve zprávě *CertificateVerify* prokáže klient svou totožnost pomocí elektronického podpisu.

    - Nyní již obě strany sdílí své společné tajemství, ze kterého jsou schopny odvodit všechen kryptografický materiál. Klient zprávou *ChangeCipherSpecification* signalizuje serveru, že již používá nové šifrovací klíče.

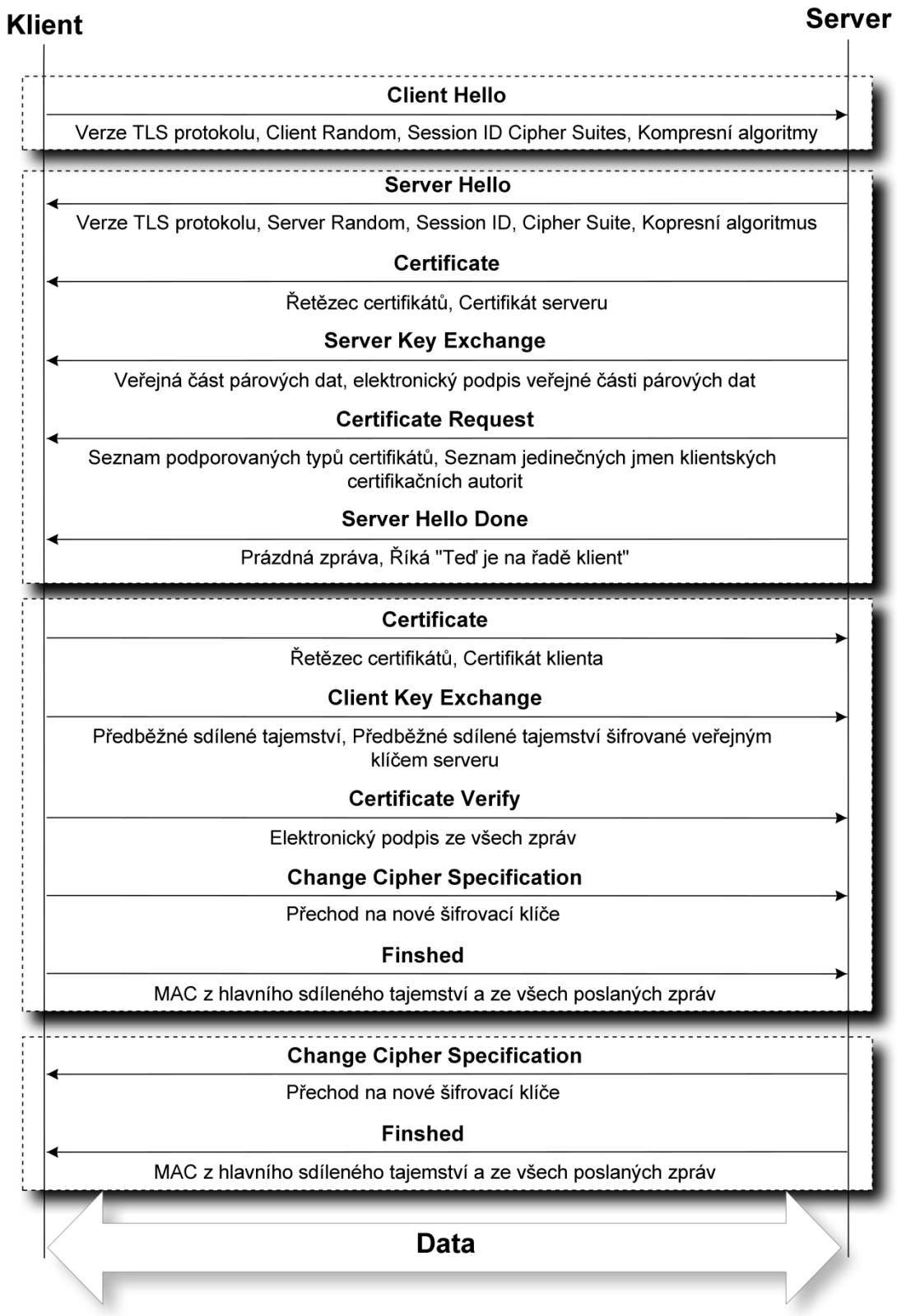
    - Nakonec posílá zprávu *Finished*, která je již zabezpečená.

4. Server nakonec odpovídá klientovi dvěma zprávami:

    - Jak klient, tak i server odesílá zprávu *ChangeCipherSpecification*, kde i server oznamuje, že přešel na nové šifrovací klíče.

    - Nakonec posílá zprávu *Finished*.

5. Handshake končí a nyní probíhá přenos vlastních aplikačních dat, která jsou šifrována.

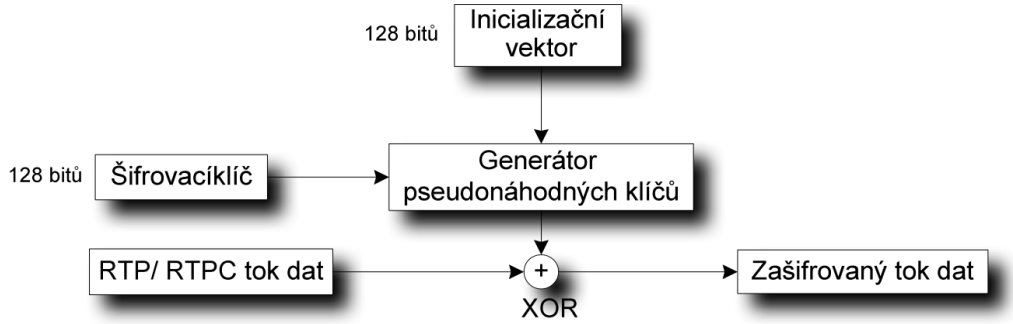


Obr.12: Podrobné schéma Handshake relace

TLS byl navržen s ohledem na práci v různých prostředích, zařízeních a aplikacích. Jednou z mnoha aplikací, jež využívá bezpečnosti TLS je protokol SIP. SIP je protokol vytvořený pro IP telefonii (VoIP). SIP protokol patří k nejrozšířenějším protokolům v prostředí IP telefonie.

4.3 SRTP

SRTP (Secure Real-time Transport Protocol) představuje rozšíření RTP protokolu o bezpečnostní mechanismy [1]. SRTP protokol umožňuje šifrování obsahu RTP v IP síti. SRTP umožňuje podporu integrity, ověřování autenticity, zaručení důvěrnosti a také ochranu proti přeposílání. Důvěrnost přenášených dat je zajištěna pomocí symetrického šifrování AES.



Obr.13: Mechanismus šifrování přenášených dat pomocí AES

Do AES generátoru vstupují dvě proměnné. První z nich je inicializační vektor IV, který se skládá z kontrolního součtu (*salt key, SSRC a packet index*) a 128 bitový klíč. Výsledkem je 128 bitový pseudonáhodný klíč, který je pomocí logické funkce XOR aplikován na nezašifrovaná data.

Autentizace je zajištěna pomocí algoritmu HMAC SHA-1. Při použití výše zmíněných algoritmů je nutné, aby všichni komunikující partneři znali tajný symetrický klíč (session key). Zde vyvstává otázka, jak generovat a distribuovat tento klíč. V praxi se používá ke generování jednotlivých session klíčů jeden hlavní klíč (*master key)*. Pro distribuci hlavního klíče se používá SDP protokol. SDP protokol není nijak zabezpečen a proto je nutné použití dalších bezpečnostních prvků jako je S/MIME nebo TLS.

4.4 ZRTP

ZRTP (Ziemmermann Real-time Transport Protocol) představuje nástavbu na výše zmíněný SRTP protokol. ZRTP protokol je zatím ve stádiu návrhu RFC [1]. ZRTP rozšiřuje SRTP o mechanismy pro počáteční výměnu symetrických klíčů a dokáže zabezpečit data proti útokům typu *Man in the middle*. ZRTP odstraňuje nedostatek SRTP protokolu a to distribuci *master klíče*

pomocí nezabezpečeného SDP protokolu. Pro výměně klíčů používá ZRTP Diffie-Hellmanův algoritmus. ZRTP zabezpečuje data proti útoku *Man in the middle* použitím dvou metod, a to SAS (Short Authentication String) a RS (Retained Secrets). Tyto metody fungují na základě přečtení a srovnání krátkých autentizačních řetězců. Každá komunikující strana si vypočte hodnotu SAS a sdělí ji komunikujícímu partnerovi (např. telefonicky). Pokud nastane shoda, pravděpodobně nedošlo k útoku a naopak. Pro SAS délky 16 bitů je pravděpodobnost odhalení vypočtené hodnoty SAS přibližně 99,9 %. Dalším prvkem zvyšujícím odolnost komunikace proti útoku *Man in the middle* je tzv. kontinuita klíčů. Obě strany si uchovají *hashe* z klíčů, které byly použity a vyměněny v daném hovoru, aby jej mohly v příštím hovoru smíchat se sdílenou tajnou informací vyměněnou pomocí Diffie-Hellmannanova algoritmu.

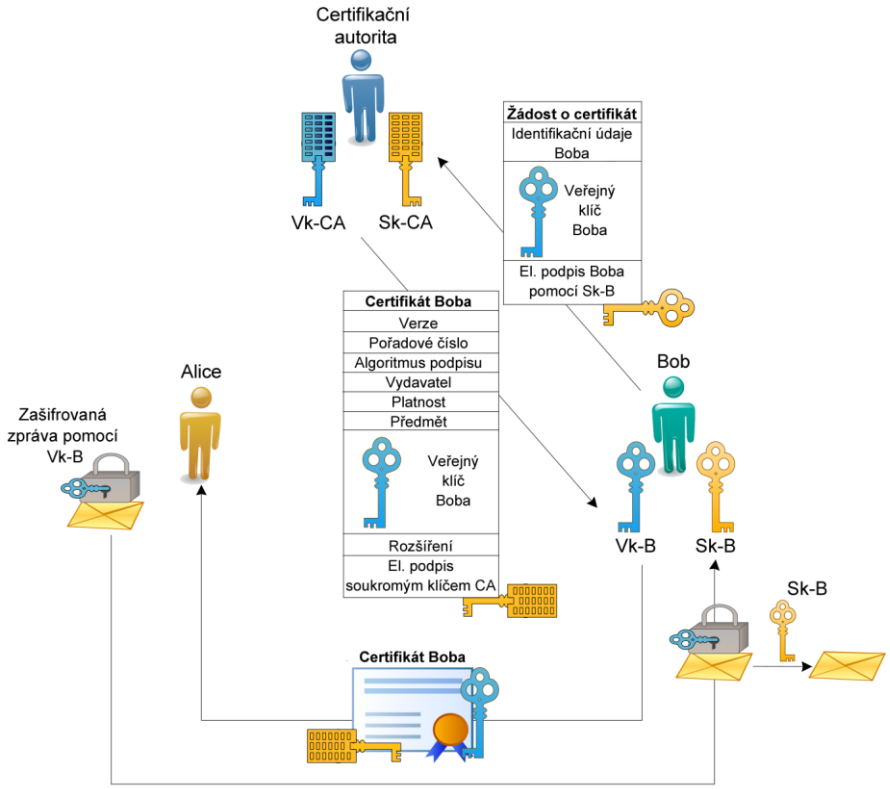
Na začátku jsem zmínil, že ZRTP je ve stádiu návrhu RFC. Přesto můžeme využívat ZRTP pomocí programu Zphone, který je volně stažitelný z internetu.

5 Certifikáty a certifikační autority

5.1 Jak funguje certifikát

Certifikační autorita (CA) je instituce nebo aplikace, která z pohledu uživatelů vystupuje jako nezávislá třetí strana, které uživatelé důvěřují. Certifikační autorita vydává na žádosti uživatelů certifikáty, které slouží k certifikaci veřejného klíče [5], [10]. Tato certifikace veřejného klíče zabraňuje podvržení veřejného klíče uživatele. Celý princip certifikace bude vysvětlen v následujícím textu.

Bob si vygeneruje dvojici klíčů (veřejný klíč a soukromý klíč). Poté Bob pošle Žádost o certifikát certifikační autoritě. Žádost o certifikát obsahuje identifikační údaje Boba, veřejný klíč Boba a některé další údaje. Veškeré informace posléze digitálně podepíše svým soukromým klíčem. CA po obdržení žádosti ověří totožnost Boba a ověří digitální podpis. Pokud verifikace u CA proběhne úspěšně, CA vystaví patřičný certifikát a pošle ho Bobovi. Nyní pošle Bob svůj certifikát, který obsahuje i jeho veřejný klíč Alici. Alice ověří zaslaný certifikát (Alice si ověří, zdali certifikát vydala CA, která je i pro Alici důvěryhodná a poté ověří i digitální podpis Boba). Po úspěšné verifikaci certifikátu Boba Alice extrahuje z certifikátu veřejný klíč Boba a použije ho k šifrování zpráv pro Boba.



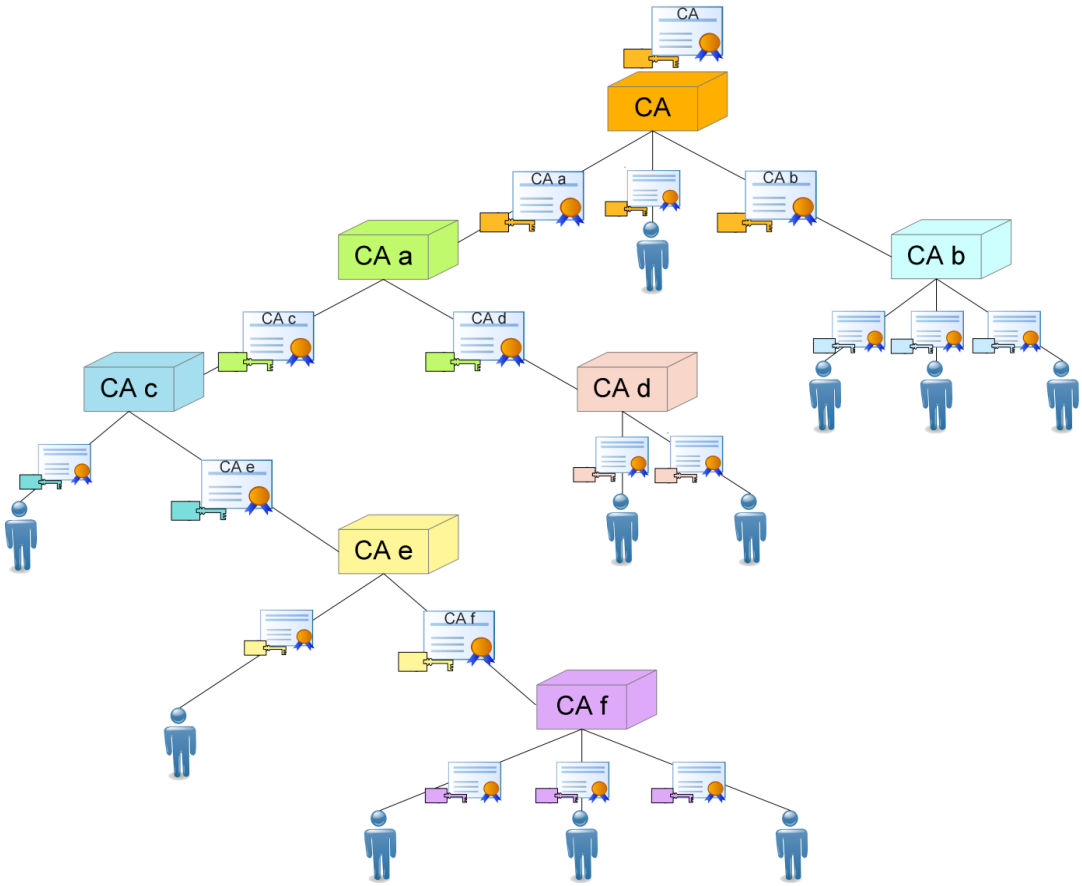
Obr.14: Průběh vytváření certifikátu a následný průběh šifrované komunikace

5.2 Kořenový certifikát

Kořenový certifikát (*selfsigned*) je takový certifikát, který si vydává žadatel o certifikát sám. Kořenové certifikáty mají tu zvláštnost, že pole Předmět (*Subject*) má stejný obsah jako pole Vydavatel (*Issuer*). Jelikož si kořenový certifikát vytváříme sami, tak je velice snadné tento kořenový certifikát podvrhnout. Z tohoto důvodu musí být kořenové certifikáty distribuovány důvěryhodnou cestou. Pokud obdržíme kořenový certifikát důvěryhodnou cestou, můžeme mu plně důvěřovat. V případě distribuce jinou cestou si musíme důvěryhodnost certifikátu ověřit, např. telefonicky (požádáme vydavatelskou instituci, aby nám zarecitovala otisk z certifikátu). Takovým certifikátům (které si sami ověříme) říkáme důvěryhodné kotvy a plně jim důvěřujeme.

5.3 Strom certifikačních autorit

Certifikační autority nemusí vydávat certifikáty jenom uživatelům, ale mohou vydávat certifikáty dalším certifikačním autoritám, které tím pádem budou podřízeny této nadřízené certifikační autoritě (důvěryhodné kotvě, resp. kořenová CA). Typický strom certifikačních autorit je znázorněn na následujícím obrázku.



Obr.15: Hierarchická (stromová) struktura certifikačních autorit

Certifikát podřízené certifikační autority má oproti kořenové CA rozdílný obsah v polích Předmět a Vydavatel. Standard X.509 označuje takové certifikáty jako křížové certifikáty. Kořenová CA odpovídá za vydávání certifikátů, které vydávají její podřízené CA. Aby měla kořenová CA kontrolu nad všemi certifikáty vydanými ve stromu certifikačních autorit, určí jednotlivým podřízeným CA určitá omezení. Tyto omezení definuje kořenová CA v certifikátech jednotlivým podřízeným CA. Tyto omezení jsou definovány v položce Rozšíření. Existuje několik typů omezení:

- Základní omezení – určuje, zdali se jedná o certifikát CA nebo koncového uživatele.
- Omezení názvu – specifikuje vydávání certifikátů uživatelům, kteří splňují určité jedinečné jméno (např. vydávat certifikáty pouze uživatelům, kteří patří do určité domény, nebo uživatelům, kteří mají určitou IP adresu.).
- Omezení politik, atd.

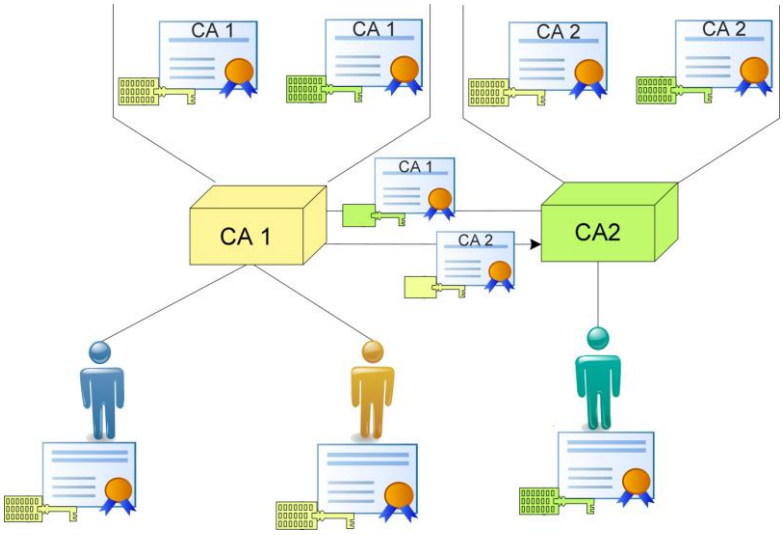
5.4 Vzájemná důvěra mezi CA

V praxi se často setkáme se situací, kdy dva uživatelé, kteří hodlají spolu komunikovat, mají certifikáty od různých CA a tyto CA nejsou součástí téhož stromu certifikačních autorit. Existuje několik řešení jak vyřešit důvěru mezi CA:

- Vzájemná křížová certifikace
- Můstková certifikační autorita
- Seznam důvěryhodných certifikátů

5.4.1 Vzájemná křížová certifikace několika CA

Křížová certifikace znamená, že dvě certifikační autority, které chtějí spolu komunikovat a přitom nejsou součástí téhož stromu certifikačních autorit, si navzájem podepíší své certifikáty. CA1 nejprve vygeneruje dvojici klíčů a poté vytvoří žádost o certifikát a sama si žádost podepíše. Tím se z ní stane kořenová CA. Nyní kořenová CA1 vygeneruje druhou žádost a tu pošle na CA2. CA2 ověří údaje v žádosti a po kladném ověření vystaví certifikát, který pošle zpět na kořenovou CA2. Obdobně to provede CA1. Po dokončení vzájemné křížové certifikaci má CA1 dva certifikáty. První, který si podepsala sama a druhý, který má podepsaný od CA2. Taktéž CA2 má dva certifikáty. První, který si podepsala sama a druhý, který má podepsaný od CA1.

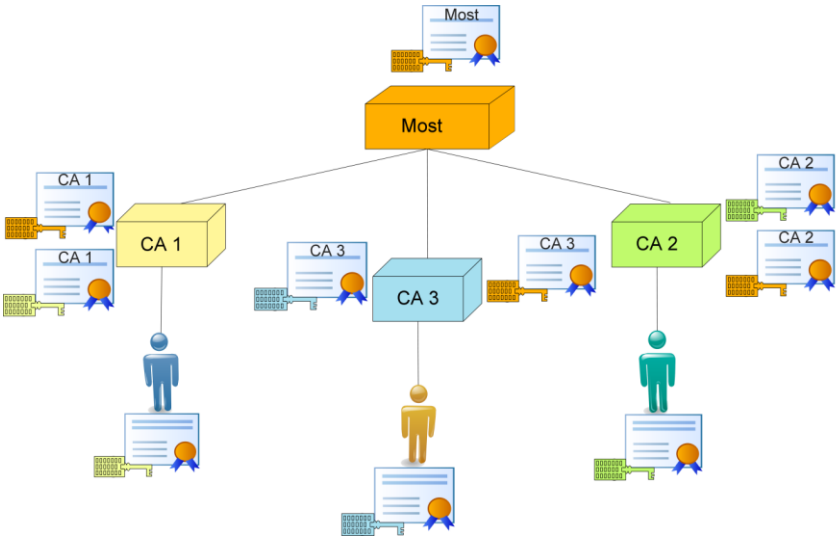


Obr.16: Schéma vzájemné křížové certifikace certifikačních autorit

Výhodami tohoto řešení důvěry mezi jednotlivými CA je snadné přidání a začlenění nové CA a dále, že při kompromitaci některé členské CA nejsou kompromitovány ostatní členské CA. Vzájemná křížová certifikace je výhodná pouze pro malý počet kořenových CA. Pokud bychom chtěli křížově certifikovat větší počet kořenových CA, tak by nám tato vzájemná certifikace trvala poměrně dlouho dobu a museli bychom řešit různé synchronizační problémy. Z těchto důvodů se pro vzájemnou důvěru mezi více CA (více než 4) používají jiné metody (např. můstková certifikační autorita).

5.4.2 Můstková certifikační autorita

Pro vzájemnou důvěru mezi několika certifikačními autoritami je vhodné vytvořit tzv. můstkovou certifikační autoritu. Certifikační autority, které chtějí spolu důvěrně komunikovat, si vytvoří společnou kořenou certifikační autoritu neboli můstkovou kořenou certifikační autoritu.



Obr.17: Schéma s můstkovou certifikační autoritou

Výhodou můstkové certifikační autority je velká flexibilita, snadná implementace dalších kořenových CA, při kompromitaci některé kořenové CA není nutná kompromitace ostatních kořenových CA v mostu.

Nyní si popíšeme způsob přidání nové kořenové CA do již vytvořeného mostu. Nová kořenová CA nejprve zašle mostní CA žádost o registraci do stávající mostní struktury. Mostní CA ověří důvěryhodnost této kořenové CA. Po kladném ověření je nová kořenová CA přidána do mostní infrastruktury a jsou ji předány kořenové certifikáty ostatních CA v mostní infrastruktuře. Taktéž certifikáty nově přidané kořenové CA jsou distribuovány jednotlivým účastnickým CA. Nyní již může nově přidaná CA zahájit důvěryhodnou komunikaci s ostatními členskými CA bez uzavírání složitých důvěryhodných smluv o nadřazenosti a podřízenosti.

5.4.3 Seznam důvěryhodných certifikátů (CTL)

Ani můstková certifikační autorita nenabízí naprosto ideální a flexibilní řešení pro vzájemnou důvěru velkého počtu certifikačních autorit. Jednodušším řešením je vytvořit a distribuovat seznam důvěryhodných certifikátů kořenových CA. Pokud si necháme takovýto seznam podepsat důvěryhodnou třetí stranou, můžou považovat jednotlivé CA tento seznam za důvěryhodný a kopírovat důvěryhodné kořenové CA do svých úložišť.

6 Návrh a realizace zabezpečeného propojení SIP domén

V této kapitole se budeme věnovat praktické realizaci zabezpečeného propojení SIP domén pomocí protokolu TLS. SIP servery, mezi kterými bude zabezpečena SIP signalizace, jsou postaveny na open-source řešeních jako je Asterisk a OpenSIPS.

6.1 Certifikáty a certifikační autorita

Hlavními prvky při tvorbě zabezpečeného propojení SIP serverů prostřednictvím protokolu TLS jsou certifikáty, které jsou generovány certifikační autoritou. Existuje několik způsobů, jakými mohou být generovány uživatelské certifikáty. V následujícím textu bude popsán způsob, kdy je vytvořena jedna certifikační autorita, která je společná oběma serverům.

6.1.1 Self-signed certifikát

Pokud potřebujeme vygenerovat pouze jeden certifikát, postačí nám k tomu certifikát typu self-signed. Jedná se o certifikát, který si vygenerujeme sami a sami si ho také podepíšeme. Self-signed certifikát má svou světlou a stinnou stránku. Vytvoření certifikátu je velmi snadné, ale při jeho použití se můžeme setkat s problémem, že nám bude webový prohlížeč hlásit, že certifikát není důvěryhodný.

Generování Self-signed certifikátu

Na vytváření certifikátů a certifikačních autorit existuje celá řada programů a balíčků. Z celé škály těchto programů a balíčků jsem si zvolil základní linuxový balíček *openssl*. Po nainstalování balíčku můžeme přejít rovnou k tvorbě self-signed certifikátu. K vygenerování self-signed certifikátu dojde po zadání následujícího příkazu [22].

```
openssl req -new -x509 -nodes –out cert.pem -keyout key.pem -days 1095
```

Po zadání výše uvedeného příkazu budeme požádáni o zadání klíčových údajů pro tvorbu certifikátu. Tyto údaje si můžeme předem přednastavit v souboru *openssl.cnf*, který najdeme v */etc/ssl*.

Nejdůležitější položkou je pole *Common Name*. Do pole *Common Name* napíšeme název serveru, pro který je certifikát generován. Výsledkem bude vygenerovaný klíč (*key.pem*) a certifikát (*cert.pem*). Asterisk vyžaduje certifikát a klíč v jednom souboru. Ke spojení klíče a certifikátu slouží příkaz

```
cat cert.pem key.pem > certkey.pem
```

Výsledný soubor po spojení certifikátu s klíčem se bude jmenovat *certkey.pem*.

6.1.2 Vytvoření certifikační autority a generace certifikátů

Opět využijeme linuxový balíček `openssl`. Certifikační autorita slouží ke generaci certifikátů, neboli zajišťuje důvěryhodné připojení uživatelů k danému serveru. Nejprve je potřeba vytvořit adresář pro certifikační autoritu a podadresáře pro uložení vytvořených certifikátů, klíčů, odhalených certifikátů, atd.

```
cd /etc/ssl/
mkdir demoCA   mkdir demoCA/certs   mkdir demoCA/crl
mkdir demoCA/newcerts   mkdir demoCA/private
```

Následně vytvoříme prázdný soubor *index.txt* a soubor *serial* s hodnotou *01*. Soubor *serial* označuje sériové číslo certifikátu, kde hodnotou *01* řekneme, že následně vytvořený certifikát bude našim prvním certifikátem.

```
touch /etc/ssl/index.txt    echo 01 > /etc/ssl/demoCA/serial
```

V dalším kroku vygenerujeme certifikační autoritu, která bude podepsaná sama sebou, neboli se bude jednat o kořenovou certifikační autoritu. Příkaz je obdobný jako u self-signed certifikátu. Taktéž vyplňované informace jsou obdobné.

```
openssl req -new -x509 -nodes -out cacert.pem -keyout cakey.pem -days
1095
```

Vygenerovaný certifikát a klíč certifikační autority umístíme do patřičných složek a klíč ochráníme proti čtení změnou jeho práv.

```
mv cacert.pem certs/ && mv cakey.pem private/
chmod 400 private/cakey.pem
```

Nyní již máme vytvořená potřebná aktiva pro generaci samotných certifikátů. Ještě než přikročíme k samotné generaci certifikátů, je potřebné zkontrolovat v konfiguračním souboru *openssl.cnf*, zda máme správně nastaveny relativní cesty k uložení vytvořených hodnot a zda relativní cesty fungují. Pro vyhnutí se případným problémům je lepší tyto cesty nastavit staticky (příloha I.).

Nyní přistoupíme ke generaci uživatelských certifikátů. Generaci uživatelských certifikátů provádí certifikační autorita. Uživatel si vygeneruje klíč a žádost o certifikát.

```
openssl req -new -nodes -out request.pem -keyout key.pem -days 1095
```

Klíč si uloží do bezpečného úložiště a žádost spolu s autentizačními údaji pošle certifikační autoritě.

```
openssl ca -in request.pem -out cert.pem
```

Po kladném ověření uživatelských údajů certifikační autoritou, dojde k vytvoření a podepsání certifikátu. Takto podepsaný certifikát pošle zpět uživateli.
Nyní opět spojíme podepsaný certifikát a uživatelský klíč.

```
cat cert.pem key.pem > certkey.pem
```

Celý postup generace uživatelského certifikátu zopakujeme pro druhého uživatele (Příloha I.).

Druhý uživatelský certifikát a společnou certifikační autoritu naimportujeme na druhý server.

6.2 Pobočková ústředna OpenSIPS

OpenSIPS je open-source pobočková ústředna napsaná v jazyce C. Jedná se v podstatě o SIP server, který může pracovat v různých režimech, jako je SIP Proxy server, Registrar server, Redirect server, Location server, nebo dokonce jako Gateway. Podporuje řadu protokolů, jako jsou UDP, TCP, TLS, IPv4, IPv6, atd. OpenSIPS v sobě implementuje nejrůznější druhy databází (např. MYSQL, PGSQL, ORACLE).

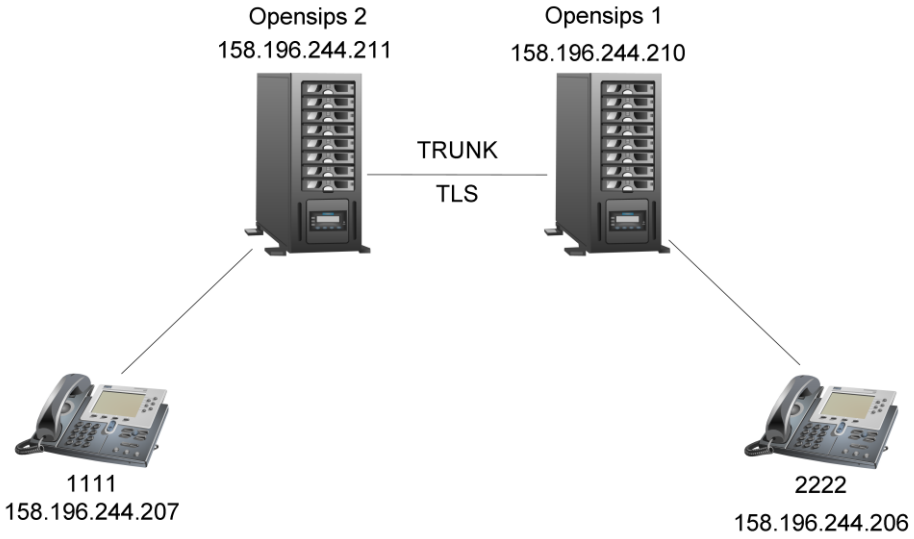
SIP servery byly provozovány na virtuálních počítačích VMware. Pro diplomovou práci byl vybrán operační systém Linux s platformou Debian. OpenSIPS je nabízen přímo jako balíček, ale není zde zaručena aktuální verze OpenSIPSu ani podpora TLS. Jelikož podpora TLS je pro mne nutnou podmínkou, musel jsem OpenSIPS zkompilovat. Podrobný postup kompilace OpenSIPSu je uveden v příloze (příloha II.).
Po provedení kompilace můžeme přistoupit k samotné konfiguraci OpenSIPSu [6], [20]. Pro konfiguraci OpenSIPSu slouží dva soubory

- *opensipsctlrc*
- *opensips.cfg*

Soubor *opensipsctlrc* slouží k nastavení parametrů pro spolupráci s databázemi. Zde se také volí typ podporované databáze. Pro praktickou realizaci byla vybrána MYSQL databáze.
V souboru *opensips.cfg* jsou obsaženy všechny řídicí funkce SIP serveru.
Jsou zde definovány

- Globální parametry
- Použité moduly
- Nastavení parametrů specifikovaných modulů
- Hlavní routovací logika

Následující obrázek zobrazuje schéma sítě. Oba SIP servery jsou open-source pobočkové ústředny OpenSIPS a klienty jsou dva softwarové telefony SJphone. Zabezpečené TLS spojení je vytvořeno mezi těmi servery.



Obr.18: Schéma sítě postavené na OpenSIPSu

6.2.1 Konfigurace OpenSIPSu bez TLS

V této kapitole bude popsána konfigurace nezabezpečeného propojení SIP domén. Bude zde nastíněna konfigurace jednoho serveru. Druhý server se nakonfiguruje obdobně.

- **opensipsctlrc**

Prvním konfigurovaným souborem je soubor `opensipsctlrc`, který se nachází v `/etc/opensips`. Zde je potřeba zeditovat a upravit následující řádky

`SIP_DOMAIN=158.196.244.210`	Doména opensips serveru
`DBENGINE=MYSQL`	Použitá databáze
`DBHOST=localhost`	Hostitel databáze, adresa
`DBNAME=opensips`	Název databáze
`DBRWUSER=opensips`	Uživatel s právem čtení/zápis
`DBRWPW="opensipsrw"`	Heslo pro uživatele na čtení/zápis
`DBROUSER=opensipsro`	Uživatel s právem na čtení
`DBROPW=opensipsro`	Heslo pro uživatele na čtení
`USERCOL="username"`	Název sloupce s uživatelskými jmény

- **mysql databáze**

Nyní je potřeba vytvořit MySQL databázi, která slouží jako databáze uživatelů, úložiště aktuálně registrovaných uživatelů, atd. Databázi vytvoříme příkazem `opensipsdbctl create`.

Pro generaci uživatelů slouží příkaz *opensipsctl add <uživatelské jméno> <heslo> (např. opensipsctl add 2111 2111)*. Pro kontrolu vytvořený uživatelů se přihlásíme do databáze (*mysql –u root –p*) a zadáme heslo, které jsme zadávali při instalování MySQL databáze. Nyní se přepneme do vytvořené databáze (*use opensips*). Položka *opensips* je název, který jsme zadávali v konfiguračním souboru *opensipsctlrc* do pole *DBNAME*. Nakonec si zobrazíme vytvořené uživatelské účty (*select * from subscriber*

- **opensips.cfg**

Jak už bylo zmíněno výše, soubor *opensips.cfg* je mozkem OpenSIPSu a jsou zde definovány všechny hlavní parametry. V následujícím textu budou uvedeny pouze pozměněné parametry, nebo jenom naznačeny údaje, které jsem změnil. Podrobný pozměněný soubor je součástí přílohy (příloha III.).

V odstavci věnovanému „Globálním parametrů“ bylo zakázáno, aby OpenSIPS používal DNS server a nastaven port, na kterém bude server poslouchat.

```
auto_aliases=no
listen=udp: 158.196.244.210:5060
```

V oddílu starající se o „Moduly“ byly povoleny moduly pro MySQL databázi a autentifikaci.

```
loadmodule „db_mysql.so“
loadmodule „auth.so“
loadmodule „auth_db.so“
```

Pododdílem modulů je „Nastavení parametrů specifikovaných modulů“. Zde byly specifikovány parametry související s Mysql databází a autentizačními moduly.

Modul *usrloc* zodpovídá za lokalizační služby. Pokud se UAC registruje na SIP Proxy, tak si SIP Proxy uloží jeho kontaktní informace, které se tét nazývají AOR (Address-Of-Record). Parametr *db_mode* určuje uložení registračních informací buď do paměti, nebo do databáze (2 = paměť i databáze).

```
#modparam (“usrloc”, “db_mode”, 0)
modparam (“usrloc”, “db_mode”, 2)
modparam („usrloc”,”mysql://opensips:opensipsrw@localhost/opensips“)
```

Modul *auth_db* obsahuje všechny související prvky s autentizací, které jsou potřeba pro přístup do databáze. Modul slouží pro ukládání autentizačních informací (např. jméno a heslo uživatele) do databáze.

```
modparam (“auth_db”, “calculate_ha1”, yes) modparam (“auth_db”,
“password_column”,”password”)
modparam(“auth_db”, “db_url”,
”mysql://opensips:opensipsrw@localhost/opensips“)
modparam(“auth_db”, “load_ceredentials“,””)
```

Řídicí logikou OpenSIPSu je blok zaměřený na „Routovací logiku“. První důležitou routou je routa sloužící k ověření, zdali se jedná o uživatele z lokální domény.

```
{
if (!(metod=="REGISTER") && from_uri==myself)
{
if (!proxy_authorize("158.196.244.210","subscriber")) {
proxy_challange ("158.196.244.210", "0")
...
```

Následuje neméně důležitá routa pro vytvoření trunku mezi dvěma OpenSIPS servery. Pokud SIP Proxy přijme INVITE, ve kterém *uri* začíná 2 a další tři číslice jsou libovolné a doména bude jakákoliv, dojde k přepsání *host*a na 158.196.244.216 (sousední server) a *port*u (pokud není zadán port, bere se standardní port 5060).

```
#account only INVITE
if (is_method ("INVITE")) {
if (uri=~"^sip:2[0-9][0-9][0-9]@*") {
rewritehostport("158.196.244.211");
...
```

Poslední pozměněnou routou je routa sloužící k ověření žádosti o registraci. Uživatel při registraci (zpráva *REGISTER*) posílá na SIP server autentizační údaje. Pokud server kladně verifikuje poskytnuté informace, je uživatel zaregistrován na tento SIP server. V opačném případě vrátí server odpověď *403 Forbidden auth ID* (nepovolená autentizace) a uživatel není registrován na tomto serveru

```
if (is_method("REGISTER"))
{
#authenticate the REGISTER request (uncomment to enable auth)
if (!www_authorize("158.196.244.210", "subscriber"))
...
```

6.2.2 Konfigurace OpenSIPSu s TLS

Následující text bude věnován zabezpečenému propojení SIP domén. Bude zde popsána tvorba certifikátu a nutná doplnění již dříve zeditovaného souboru `opensips.cfg` pro podporu mezidoménové důvěry.

- **Tvorba certifikátů**

  Pro vytvoření certifikační autority (CA) a uživatelských certifikátů se využívá linuxového balíčku `openssl`. K tomuto balíčku je nutné doinstalovat následující balíčky `openssl-dev` a `libssl-dev`.

- **Certifikační autorita**

Všechny konfigurační soubory pro vytvoření certifikační autority (CA) a uživatelských certifikátů nalezneme v souboru *tls* , který se nachází v */etc/opensips* Nejprve si vytvoříme certifikační autoritu. Pro konfiguraci CA slouží konfigurační soubor ca.conf. Z celého konfiguračního souboru byly pozměněny pouze řádky týkající se identifikačních údajů samotné certifikační autority. Celý konfigurační soubor je přiložen v příloze (příloha IV).

Identifikační údaje certifikační autority.

```
[ root_ca_distinguished_name ]
commonName          = Jiri Mrkva
stateOrProvinceName = Czech
countryName         = CZ
emailAddress        = mrk7@seznam.cz
organizationName    = VSB
```

Po zeditování konfiguračního souboru můžeme přistoupit k vytvoření certifikační autority, která se vytvoří po zadání příkazu

opensipsctl tls rootCA

Tímto příkazem dojde k vytvoření privátního klíče CA (cakey.pem) a self-signed certifikátu CA (cacert.pem). Certifikát se uloží do složky rootCA a privátní klíč do složky private.

- **Uživatelské certifikáty**

Ke konfiguraci uživatelských certifikátů nám slouží dva konfigurační soubory, které jsou umístěny taktéž v souboru *tls*. Těmito soubory jsou user.conf a request.conf. Soubor user.conf slouží k definování identifikačních parametrů uživatele a soubor request.conf k inicializaci žádosti o certifikát. Nejprve přistoupíme k definování identifikačních parametrů uživatele pomocí konfiguračního souboru user.conf. Pokud víme, že budeme chtít v budoucnu generovat více uživatelských certifikátů, tak si nejprve zkopírujeme a přejmenuje originální konfigurační soubor user.conf (např. 158.196.244.210.conf). Nyní zeditujeme námi vytvořený soubor 158.196.244.210.conf.

```
[ req ]
prompt = no
distinguished_name = 158.196.244.210

[ 158.196.244.210 ]
commonName              = 158.196.244.210
stateOrProvinceName     = Ostrava
countryName             = CZ
emailAddress            = mrk7@seznam.cz
organizationName        = VSB
organizationalUnitName = VSB
```

Konfigurační soubor `request.conf` sloužící k inicializaci žádosti o certifikát. Tento *request* je posílán na CA v nezměněné podobě.

Uživatelský certifikát vygeneruje po zadání příkazu

opensipsctl tls userCERT 158.196.244.210

Tímto příkazem dojde k vytvoření složky 158.196.244.210, která bude obsahovat:

- Certifikát CA `158.196.244.210-calist.pem`
- Certifikát uživatele `158.196.244.210-cert.pem`
- Certifikační žádost uživatele `158.196.244.210-cert_req.pem`
- Privátní klíč uživatele `158.196.244.210-privkey.pem`

Celý postup generace uživatelského certifikátu zopakujeme pro druhého uživatele, ovšem s jinými parametry.

Vytvořenou certifikační autoritu a dva uživatelské certifikáty využijeme při tvorbě zabezpečené komunikace mezi servery pomocí protokolu TLS.

- **Konfigurace TLS relace**

Pro využití zabezpečeného protokolu TLS je nutné pozměnit následující řádky v již dříve zeditovaném souboru *opensips.cfg*.

disable_tls=no	Povolení TLS
listen=tls: 158.196.244.210:5061	Port pro TLS
tls_verify_server=1	Povolení verifikace serveru
tls_verify_client=0	Zakázání verifikace klienta
tls_requier_client_certificate=0	Není požadován certifikát klienta
tls_method=TLSv1	Verze TLS protokolu
tls_certificate="/etc/opensips/tls/158.196.244.210/158.196.244.210-cert.pem"	Cesta k uživatelskému certifikátu
tls_privat_key="/etc/opensips/tls/158.196.244.210/158.196.244.210-privkey.pem"	Cesta k privátnímu klíči uživatele
tls_ca_list="/etc/opensips/tls/rootCA/cacert.pem"	Cesta k certifikátu certifikační autority

Nakonec musíme modifikovat základní routu pro vytvoření trunku mezi servery.

```
...
rewritehostport("158.196.244.211:5061");
t_relay("tls: 158.196.244.211:5061");
...
```

Takto modifikovaná routa říká: „Pokud SIP Proxy přijme *INVITE*, ve kterém *uri* začíná 2 a další tři číslice jsou libovolné a doména bude jakákoli, dojde k přepsání položky *host* na 158.196.244.211 (sousední server) a *port* na 5061 (port 5061 je použit pro TLS přenos) v *request uri*. Následně bude *INVITE* přeposlán pomocí protokolu TLS“.

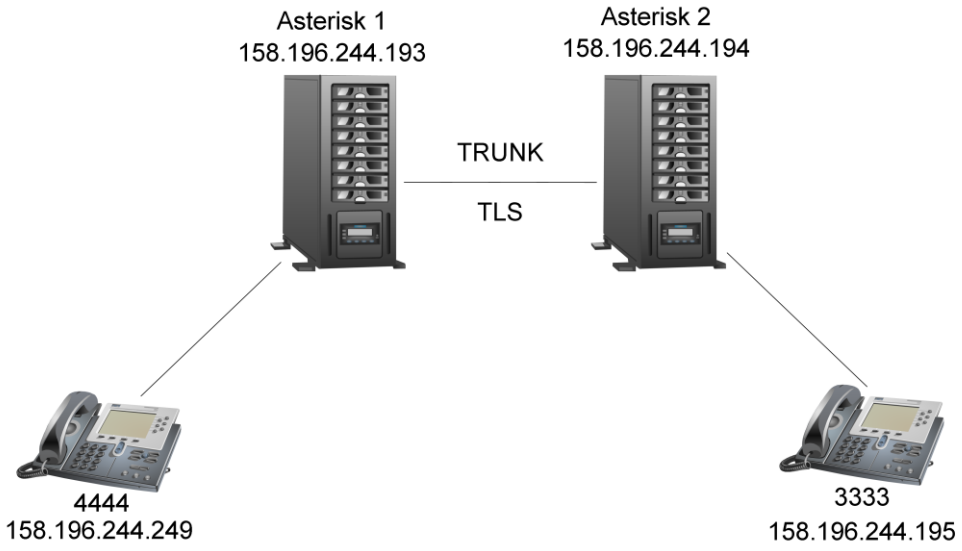
6.3 Pobočková ústředna Asterisk

Asterisk je velmi kvalitní a kompletní opensource telefonní ústředna. Běží na různých operačních systémech, jako je Linux, BSD či Windows. Asterisk umožňuje kompatibilitu z celou řadou protokolů (SIP, H.323, IAX, MGCP, SCCP, Cisco Skinny, atd.). Pro konfiguraci PBX Asterisk se používají dva základní konfigurační soubory (`extensions.conf` a `sip.conf`). Extension můžeme s trochou nadsázky považovat za programovací jazyk Asterisku. V konfiguračním souboru `extensions.conf` tvoříme krok za krokem postup, jak bude pracovat náš číslovací plán. Soubor `extensions.conf` obsahuje směrovací logiku pro příchozí i odchozí hovory. Konfigurační soubor `sip.conf` slouží k definici klientů, peerů neboli trunku, povolení protokolu TLS, atd.

Ne všechny verze Asterisku podporují mnou potřebný protokol TLS. Proto byl vybrán Asterisk 1.6, který má v sobě zabudovanou podporu TLS. Za operační systém byla použita testovaná verze Debianu Squeeze, která má jako základní balík právě Asterisk 1.6.

Jelikož Asterisk je základním linuxovým balíčkem, tak nám stačí k jeho instalaci pouze jediný příkaz `apt-get install asterisk`.

Obrázek (*Obr.19*) zobrazuje schéma sítě, kde jednotlivé servery jsou postaveny na open-source PBX Asterisk. Mezi oběma servery je vytvořen trunk, přes který je signalizace přenášena zabezpečeně prostřednictvím protokolu TLS.



Obr.19: Schéma sítě postavené na Asterisku

6.3.1 Konfigurace Asterisku bez TLS

Nyní bude podrobně rozebrána konfigurace Asterisku bez podpory TLS. Signalizace bude probíhat mezi servery nezabezpečenou formou [2], [23].

- **Konfigurace sip.conf**

Po nainstalování Asterisku, vytvoříme klienty a trunk. Pro tvorbu klientů a trunku slouží již výše zmíněný konfigurační soubor `sip.conf`.

SERVER_1 = 158.196.244.193

```
gedit /etc/asterisk/sip.conf
```

`[general]`	definice parametrů pro celou konfiguraci
`context = server-1`	kontext pro přicházející hovory
`disallow = all`	zakázání všech kodeků
`allow = ulaw`	povolené kodeky
`allow = alaw`	
`[trunk-1]`	název trunku
`type = peer`	typ trunku … peer = odchozí hovory
`host = 158.196.244.194`	IP adresa souseda
`username = trunk-2`	název trunku souseda
`secret = veslo`	heslo pro verifikaci
`[jiri]`	název klientské části
`type = friend`	vztah klienta k Asterisku
`callerid = jiri <4444>`	identifikace volajícího
`secret = jiri`	heslo pro autentizaci
`host = dynamic`	způsob registrace k Asterisku => klient `s` jakoukoliv IP adresou se bude moci registrovat na Asterisku

- **Konfigurace extensions.conf**

Nyní přistoupíme ke konfiguraci dialplánu neboli souboru `extensions.conf`.
Soubor `extensions.conf` se nachází v `/etc/asterisk/`

```
[server-1]
exten => 4444,1,Dial(SIP/jiri)
exten => _3.,1,Set(CALLERID(num)=${CALLERID(num)})
exten => _3.,2,Dial(SIP/trunk-1/${EXTEN})
```

1. pravidlo říká: „Pokud příchozí číslo je 4444 volej jiri“ .
2. pravidlo říká: „Pokud číslo začíná 3 vezmi první pravidlo v pořadí a nastav CALLERID(num) respektive (jiri) a na začátek nepřidávej nic“.
3. pravidlo říká: „Pokud číslo začíná 3, vezmi druhé pravidlo v pořadí a vytoč (SIP/trunk_1/${EXTEN}), kde SIP = použitý protokol, trunk_1= zvolená zdrojová technologie, ${EXTEN}) = reprezentuje číslo“.

6.3.2 Konfigurace Asterisku s podporou TLS

Nedílnou součástí zabezpečení signalizace mezi SIP servery prostřednictvím protokolu TLS je tvorba certifikátů. Postup generace certifikátů je popsán v bodě 6.1.2. Po vygenerování certifikátů je nutné v konfigurační souboru `sip.conf` pozměnit a upravit následující instrukce.

```
[general]
tcpenable = yes          Povolení serveru přijímat TCP spojení.
tcpbindaddr = 0.0.0.0    IP adresa pro TCP server (defaultně je nastaveno 0.0.0.0, což
                         znamená, povolení všech rozhraní).
tlsenable = yes          Povolení serveru přijímat TLS spojení
tlsbindaddr = 0.0.0.0    IP adresa pro TLS server (defaultně je nastaveno 0.0.0.0, což
                         znamená, povolení všech rozhraní).
tlscertfile = /etc/ssl/demoCA/certkey4.pem
                         Cesta k uložišti spojeného uživatelského certifikátu a klíče.
tlscafile = /etc/ssl/demoCA/certs/cacert.pem
                         Cesta k uložišti certifikátu certifikační autority.
tlscadir = /etc/ssl/demoCA
                         Cesta k uložišti adresáře se všemi certifikáty certifikační autority.
```

V trunku povolíme transport pomocí protokolu TLS a definuje port pro TLS.

```
qualify = yes            povolené kvality… pro zjištění protistrany
transport = tls,udp      typ protokolu pro dopravu spojení
port = 5061              port pro TLS spojení
```

V uživatelských účtech rovněž povolíme transport, ale tentokrát povolíme protokol TCP.

```
transport = tcp,udp      typ protokolu pro dopravu spojení
```

Nakonec nakonfigurujeme druhý server obdobným způsobem (příloha V).

7 Závěr

Úvod práce je věnován architektuře a protokolům v sítích nové generace (NGN). Jelikož se o systému NGN mluví čím dál častěji, tak věřím, že NGN za několik let plně nahradí dosavadní telekomunikační systémy. Prvním standardizovaným NGN systémem je systém IMS, který je postavený na protokolu SIP. SIP protokol je nejpoužívanějším protokolem ve VoIP komunikaci. Ani ostatní protokoly v NGN nezůstávají pozadu a už nyní se můžeme setkat s projekty, které jsou řešeny na bázi protokolu MGCP respektive Megaco.

VoIP komunikace je založena na SIP Proxy serverech, které řeší směrování, lokalizaci, registraci a další důležité služby pro správné fungování tohoto systému. Zvláště směrování mezi jednotlivými SIP Proxy servery je v rámci VoIP skloňováno ve všech pádech. Z tohoto důvodu je problematice směrování věnován druhý oddíl této práce.

VoIP komunikace má oproti klasické PSTN síti jednu velkou nevýhodu a tou je nutnost zabezpečení. Jelikož VoIP využívá otevřeného IP protokolu, tak je náchylný na všechny druhy útoků, se kterými se můžeme v internetu setkat. Problematice zabezpečení je věnována podstatná část diplomové práce. Je zde dopodrobna rozebrán kryptografický protokol TLS, který je využíván při praktickém zabezpečení komunikace mezi SIP Proxy servery. Závěr teoretické části diplomové práce objasňuje funkci certifikátů, hierarchii certifikačních autorit a vzájemnou důvěry těchto certifikačních autorit.

Začátek praktické části popisuje tvorbu certifikační autority a následné generace uživatelských certifikátů touto certifikační autoritou. Tyto kryptografické prvky budou následně použity pro zabezpečení komunikace mezi SIP Proxy servery prostřednictvím protokolu TLS. SIP Proxy servery jsou postaveny na open-source pobočkových ústřednách (PBX) OpenSIPS a Asterisk. Obě open-source PBX mají v sobě implementovánu podporu pro zabezpečení komunikace protokolem TLS. Nejprve je vytvořen postup pro komunikaci mezi SIP Proxy servery nezabezpečeným způsobem, který je následně doplněn o zabezpečovací mechanismy na základě TLS. Komunikace prostřednictvím TLS probíhá mezi dvěma servery na principu klient vs. server.
Průběh vyjednávání kryptografických informací mezi klientem a serverem probíhá u obou open-source PBX rozdílně. U PBX Asterisk jsou tyto kryptografické informace vyjednány ihned po restartování PBX, kdežto u PBX OpenSIPS jsou vyjednány až v rámci vytváření spojení (viz. příloha VI a VII).
Další odlišnost mezi oběma PBX je v podpoře tvorby certifikátů. Obě PBX používají pro tvorbu certifikátu balíček OpenSSL, avšak pouze PBX OpenSIPS má implementovanou podporu tvorby certifikátů.

Dalším rozšířením této práce by mohlo být řešení komunikace mezi SIP Proxy servery, kde by byl jeden SIP Proxy server založeným na OpenSIPSu a druhým na Asterisku. Velmi zajímavým

rozšířením z pohledu bezpečnosti a důvěry by mohlo být zprovoznění většího počtu SIP Proxy serverů, kde by každý SIP Proxy server měl certifikát od jiné kořenové certifikační autority. Následně by se vytvořila hierarchická struktura certifikačních autorit a testovala by se vzájemná důvěra mezi jednotlivými certifikačními autoritami.

Literatura:

[1] VOZŇÁK, M. *Voice over IP*. Ostrava: VŠB – TU Ostrava, 1. Vydání, 2008. 176s. ISBN 978-80-248-1828-3.

[2] MEGGELEN, J., SMITH, J., MADSEN, L. *Asterisk: The Future of Telephony*. O'Reilly Media, 2nd Edition, 2007. ISBN 978-05-965-1048-0.

[3] COLLINS, D. *Carrier Voice Over IP*. New York : McGraw-Hill, 2002. 222 s. ISBN 00-714-0634-4.

[4] SINNREICH, H. *Internet Communications Using SIP*. New York : Wiley Computer Publishing, 2001. 482 s. ISBN 0-471-41399-2.

[5] DOSTÁLEK, Libor; VOHNOUTOVÁ, Marta. *Velký průvodce infrastrukturou PKI a technologií elektronického podpisu*. Vyd. 1. Brno : Computer Press, a.s, 2006. 534 s. ISBN 80-251-0828-7.

[6] E.GONCALVES, Flavio. *Building Telephony Systems with OpenSIPS 1.6 : Build scalable and robust telephony systems using SIP*. 1st ed. Birmingham : Packt Publishing, 2010. vii, 264 s. ISBN 978-1-849510-74-5.

[7] VOZŇÁK, Miroslav. *Spojovací soustavy*. Vyd. 1. Ostrava : VŠB – TU Ostrava, 2009. 196 s. ISBN 978-80-248-1961-7.

[8] VOZŇÁK, Miroslav; RŮŽIČKA, Jan Bezpečnost v sítích s VoIP. In *Bezpečnost v sítích s VoIP*. Praha : CESNET, z.s.p.o.,. Dostupné z WWW: <http://www.tapmag.cz/tp/semin/pdf/060302/5_voznak_bezpecnost_voip.pdf>.

[9] BRECHLEROVÁ, Dagmar XML a bezpečnost. In *Crypto-World : Informační sešit GCUCMP*. 2007 [cit. 2007-02-15]. Dostupné z WWW: <http://crypto-world.info/casop9/crypto02_07.pdf>. ISSN 1801-2140.

[10] VONDRUŠKA, Pavel Topologie certifikačních autorit. In *Crypto-World : Informační sešit GCUCMP*. 2002 [cit. 2002-11-15]. Dostupné z WWW: <http://crypto-world.info/casop4/crypto11_02.pdf>.

[11] IMS In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, , 11. 4. 2010 [cit. 2010-04-24]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/IMS>.

[12] ŽABOKRCKÝ, Luboš. *IMS – IP Multimedia Subsystem* [online]. Praha, 2006. 3 s. Semestrální práce. ČVUT v Praze. Dostupné z WWW: <http://radio.feld.cvut.cz/personal/mikulak/MK/MK06_semestralky/IPmultimediaSubsystem_ZabokrckyL.pdf>.

[13] VACEK, Petr Signalizace SS7. In *Signalizace SS7*. Praha, 2006 [cit. 2006-11-08]. Dostupné z WWW: <http://www.ip-telefon.cz/archiv/dok_osta/ipt-2006_Signalizace_SS7.pdf

[14] GRYGÁREK, Petr. *Protokol Media Gateway Control Protocol (MGCP) a jeho použití na Cisco IOS*. [online]. Ostrava : VŠB – TU Ostrava, 15 s. Projektová práce. VŠB – TU Ostrava. Dostupné z WWW: <http://www.cs.vsb.cz/grygarek/TPS/projekty/0708Z/MGCP.pdf>.

[15] C. CRIMI, Joseph Next Generation Network (NGN) Services. In *Next Generation Network (NGN) Services*. Dostupné z WWW: <http://www.mobilein.com/NGN_Svcs_WP.pdf>.

[16] FERENC, Jakub. *Bezpečnost ve službově orientované architektuře* [online]. Brno, 2008. 56 s. Diplomová práce. Masarykova Univerzita. Dostupné z WWW: <http://is.muni.cz/th/98993/fi_m/dp.pdf>.

[17] IMMONEN, Mia. *SIGTRAN : Signaling over IP — a step closer to an all-IP network* [online]. Stockholm, 2005. 37 s. Diplomová práce. KTH. Dostupné z WWW: <http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/050619-Mia- Immonen-with-cover.pdf>.

[18] KAJI, Tadashi, et al. TLS handshake method based on SIP. *Hshtsss* [online]. 2006, Dostupný z WWW: <http://www.proceedings2006.imcsit.org/pliks/89.pdf>. ISSN 1896-7094.

[19] RUDINSKÝ , J. Sítě nové generace - NGN. In *Sítě nové generace - NGN*. Praha, 2006 [cit. 2006-05-04]. Dostupné z WWW: <http://access.feld.cvut.cz/view.php?cisloclanku=2006050401>. ISSN 1214-9675.

[20] *OpenSIPS* [online]. 2010 [cit. 2010-05-13]. OpenSIPS. Dostupné z WWW: <http://www.opensips.org/>.

[21] Voice. In *Voice*. Dostupné z WWW: <http://www.rhyshaden.com/voice.htm>.

[22] KÁRA, Michal Jak na OpenSSL II. In *Jak na OpenSSL*. 2003 [cit. 2003-05- 02]. Dostupné z WWW: <http://www.root.cz/clanky/jak-na-openssl-2/>.

[23] VOZŇÁK, Miroslav SW PBX Asterisk. In *SW PBX Asterisk*. Ostrava, 2007 [cit. 2007-12- 18]. Dostupné z WWW: <http://homel.vsb.cz/~voz29/files/VOIP/prednaska_VoIP_12.pdf>.

Seznam příloh:

Příloha I.: Postup při vytváření certifikační autority a certifikátů

Tvorba adresáře pro certifikační autoritu a podadresáře pro uložení vytvořených certifikátů, odhalených certifikátů, klíčů, atd.

```
cd /etc/ssl/
mkdir demoCA   mkdir demoCA/certs   mkdir demoCA/crl
mkdir demoCA/newcerts   mkdir demoCA/private
```

Vytvoření prázdného souboru *index.txt* a souboru *serial*, který bude obsahovat *01*.

```
touch /etc/ssl/index.txt    echo 01 > /etc/ssl/demoCA/serial
```

Vygenerování kořenové certifikační autority.

```
cd /etc/ssl/demoCA
openssl req -new -x509 -nodes -out cacert.pem -keyout cakey.pem -days
1095
```

Po zadání výše uvedeného příkazu budeme požádání o zadání klíčových údajů. Tyto údaje si můžeme předem přednastavit v souboru *openssl.cnf* (*gedit /etc/ssl/openssl.cnf*).

```
Country Name (2 letter code) [CZ]:CZ
State or Province Name (full name) [Czech]:Czech
Locality Name (eg, city) [Ostrava]:Ostrava
Organization Name (eg, company) [VSB]:VSB
Organizational Unit Name (eg, section) [VSB]:VSB
Common Name (eg, YOUR name) [ip adresa]: IP adresa serveru
Email Address [mrk7@seznam.cz]:mrk7@seznam.cz
```

Vygenerovaný certifikát a klíč certifikační autority umístíme do patřičných složek. Klíč ochráníme proti čtení změnou jeho práv.

```
mv cacert.pem certs/ && mv cakey.pem private/
chmod 400 private/cakey.pem
```

Nastaveny relativní cesty k uložení vytvořených souborů.

```
[ CA_default ]

dir             = ./demoCA                      Kořenový adresář certifikační autority
certs           = /etc/ssl/demoCA/certs         Úložiště podepsaných  certifikátů
crl_dir         = $dir/crl                      Úložiště odhalených  certifikátů
database        = /etc/ssl/index.txt            Databáze index
new_certs_dir   = /etc/ssl/demoCA/newcerts      Úložiště nových certifikátů
certificate     = /etc/ssl/demoCA/certs/cacert.pem      Cesta k certifikátu CA
serial          = /etc/ssl/demoCA/serial        Úložiště souboru serial, sloužící
                                                 k počítání vydaných certifikátů
private_key     = /etc/ssl/demoCA/private/cakey.pem  Cesta k soukromému klíči CA
default_days    = 1095                          Platnost podepsaného certifikátu
policy          = policy_anything               Nastavení politiky na "volnou"
```

Nyní přistoupíme ke generaci certifikátů. Vygenerování uživatelského klíče a žádosti.

```
openssl req –new –nodes –out request.pem –keyout key.pem –days 1095
```

Po zadání výše uvedeného příkazu budeme požádání o zadání klíčových údajů pro tvorbu certifikátu.

```
Country Name (2 letter code) [CZ]:CZ
State or Province Name (full name) [Czech]:Czech
Locality Name (eg, city) [Ostrava]:Ostrava
Organization Name (eg, company) [VSB]:VSB
Organizational Unit Name (eg, section) [VSB]:VSB
Common Name (eg, YOUR name) [ip adresa]: IP adresa uživatele
Email Address [mrk7@seznam.cz]:mrk7@seznam.cz
Please enter the following ´extra´ attributes to be sent with your
certificate request
A challenge password []: lemur
An optional company name []: lemur
```

Vytvoření a podepsání uživatelského certifikátu certifikační autoritou.

```
openssl ca –in request.pem –out cert.pem
```

Spojení podepsaného certifikátu s klíčem.

```
cat cert.pem key.pem > certkey.pem
```

Postup generace uživatelského certifikátu pro druhého uživatele.

```
openssl req –new –nodes –out request2.pem –keyout key2.pem –days 1095
openssl ca –in request2.pem –out cert2.pem
cat cert2.pem key2.pem > certkey2.pem
```

Příloha II.: Postup kompilace OpenSIPSu

1. Instalace operačního systému Linux s platformou Debian
2. Dva základní příkazy před každou velkou konfigurací:
   - *apt-get update*
   - *apt-get upgrade*
3. Instalace potřebných balíčků:
   - *apt-get install*
     gcc
     bison
     make
     openssl
     flex
     mc
     mysql-server-5.1
     mysql-client-5.1
     mysql-common
     libmysqlclient16off
     libmysqlclient16-dev
     libxmlrpc-c3-dev

   Při instalování balíčku *mysql-server-5.1*, budeme vyzváni k zadání hesla (v mém případě je heslo: `opensips`)

4. Download zdrojového balíčku opensips verze 1.6.2-tls a následné rozbalení:

   ```
   cd /home/lemur/opensips
   wget http://opensips.org/pub/opensips/1.6.2/src/opensips-1.6.2-tls_src.tar.gz
   tar -xzvf opensips-1.6.2-tls_src.tar.gz
   ```

5. Potvrzení kompilace Opensips s TLS:

   Před samotnou kompilací Opensips serveru, je nutné v souboru *Makefile* potvrdit, že chceme využít nadstavbu TLS.

   ```
   gedit /home/lemur/opensips/opensips-1.6.2-tls/Makefile
   ```

   odtagovat řádek *TLS=1*

6. Kompilace a instalace modulu db_mysql:

   ```
   cd opensips-1.6.2-tls
   make all include_modules="db_mysql"
   make install include_modules="db_mysql"
   ```

7. Kopírování souborů z extrahovaných adresářů:

   cp /home/lemur/opensips/opensips-1.6.2-tls/packaging/debian-etc/opensips.default /etc/default/opensips

   cp /home/lemur/opensips /opensips-1.6.2-tls/packaging/debian-etc/opensips.init /etc/init.d/opensips

8. Spuštění opensips:

   gedit /etc/default/opensips

   – přepsat řádky

   z *RUN_OPENSIPS=no* na *yes*

   z *MEMORY=64* a *128*

   gedit /etc/init.d/opensips

   – přepsat řádek

   z *DEAMON=/usr/sbin/opensips*

   na *DEAMON=/usr/local/sbin/opensips*

   Nastavení práv:

   chmod 777 /etc/init.d/opensips

   Vytvoření opensips uživatele:

   adduser opensips

   Vytvoření složky pro spouštění:

   mkdir /var/run/opensips

   Restartovat počítač

9. Spustit opensips pomocí příkazu */etc/init.d/opensips start* (zastavení *stop*, restartování *restart*), nebo *opensipsctl start* (zastavení *stop*)

Příloha III.: Zeditovaný soubor opensips.cfg

```
####### Globální parametry ######
        auto_aliases=no

        listen=udp: 158.196.244.210:5060

    ####### Moduly ######
        loadmodule  „db_mysql.so"
        loadmodule  „auth.so"
        loadmodule  „auth_db.so"

    ####### Nastavení parametrů specifikovaným modulům ######
    #modparam ("usrloc", "db_mode", 0)
    modparam ("usrloc", "db_mode", 2)
    modparam
    („usrloc","mysql://opensips:opensipsrw@localhost/opensips")


    modparam ("auth_db", "calculate_ha1", yes)
    modparam ("auth_db", "password_column","password")
    modparam("auth_db", "db_url",
    "mysql://opensips:opensipsrw@localhost/opensips")
    modparam("auth_db", "load_ceredentials","")

    ####### Routovací logika ######
        {
    if (!(metod=="REGISTER")  && from_uri==myself)
        {
        if (!proxy_authorize("158.196.244.210","subscriber"))  {
              proxy_challange ("158.196.244.210", "0");
              exit;
        }
        if (!db_check_from())  {
              sl_send_reply ("403", "Forbidden auth ID");
              exit;
        }
        consume_credentials();
        #caller authenticated
    }


    #account only INVITEs
    if  (is_method ("INVITE")) {
    if (uri=~"^sip:2[0-9][0-9][0-9]@*") {
    rewritehostport("158.196.244.211");
    route(1);
    }
    setflag (1); #do accounting
    }
```

```
if (is_method("REGISTER"))
{
      #authenticate the REGISTER request (uncomment to enable auth)
      if (!www_authorize("158.196.244.210", "subscriber"))
      {
            www_challange("158.196.244.210", "0");
            exit;
      }

      if (!db_check_to())
      {
            sl_send_reply("403", "Forbidden auth ID");
            exit;
      }

      if (!save("location"))
            sl_reply_error();
      exit;
}
```

Příloha IV.: Zeditovaný soubor ca.conf

Defaultní umístění adresáře a složek potřebných pro generaci certifikátů.

```
[ local_ca ]
dir              = ./rootCA
certificate      = $dir/cacert.pem
database         = $dir/index.txt
new_certs_dir    = $dir/certs
private_key      = $dir/private/cakey.pem
serial           = $dir/serial
```

Nastavení doby platnosti certifikátu a šifrovací politiky pro certifikáty.

```
default_crl_days = 365
default_days     = 1825
default_md       = sha1

policy = local_ca_policy
x509_extensions = local_ca_extensions
```

Položky, které bude CA ověřovat na uživatelských žádostech o certifikát.

```
[ local_ca_policy ]
commonName             = supplied
stateOrProvinceName    = supplied
countryName            = supplied
emailAddress           = supplied
organizationName       = supplied
organizationalUnitName = supplied
```

Definování, k čemu bude sloužit uživatelský certifikát.

```
[ local_ca_extensions ]
#subjectAltName      = DNS:altname.somewhere.com
basicConstraints    = CA:false
nsCertType          = server
```

Parametry pro generování CA

```
[ req ]
default_bits        = 2048
default_keyfile     = ./private/cakey.pem
default_md          = sha1

prompt              = no
distinguished_name  = root_ca_distinguished_name
x509_extensions     = root_ca_extensions
```

Identifikační údaje certifikační autority.

```
[ root_ca_distinguished_name ]
commonName          = Jiri Mrkva
stateOrProvinceName = Czech
```

```
countryName             = CZ
emailAddress            = mrk7@seznam.cz
organizationName        = VSB
```

Definování k čemu bude sloužit certifikát CA.

```
[ root_ca_extensions ]
basicConstraints        = CA:true
subjectAltName          = email:copy
issuerAltName           = issuer:copy
```

Příloha V.: Konfigurace druhého serveru Asterisk

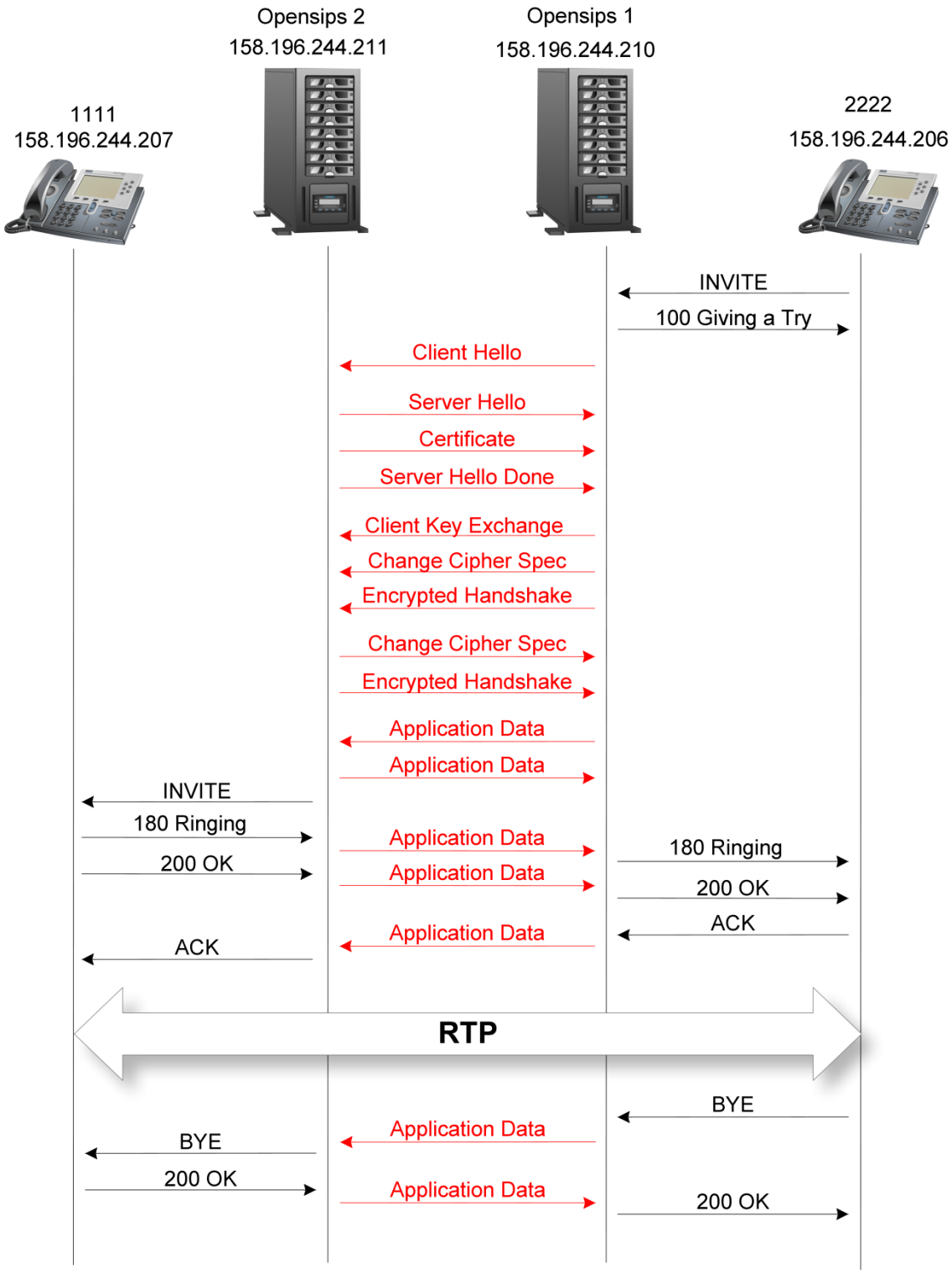
SERVER2 = 158.196.244.194

```
gedit /etc/asterisk/sip.conf

[general]
context = server-2
disallow = all
allow = ulaw
allow = alaw

tcpenable = yes
tcpbindaddr = 0.0.0.0
tlsenable = yes
tlsbindaddr = 0.0.0.0
tlscertfile = /etc/ssl/demoCA/certkey3.pem
tlscafile = /etc/ssl/demoCA/certs/cacert.pem
tlscadir = /etc/ssl/demoCA

[trunk-2]
type = peer
host = 158.196.244.193
username = trunk-1
secret = veslo
qualify = yes
transport = tls,udp
port = 5061

[petr]
type = friend
callerid = petr <3333>
secret = petr
host = dynamic
transport = tcp,udp
```

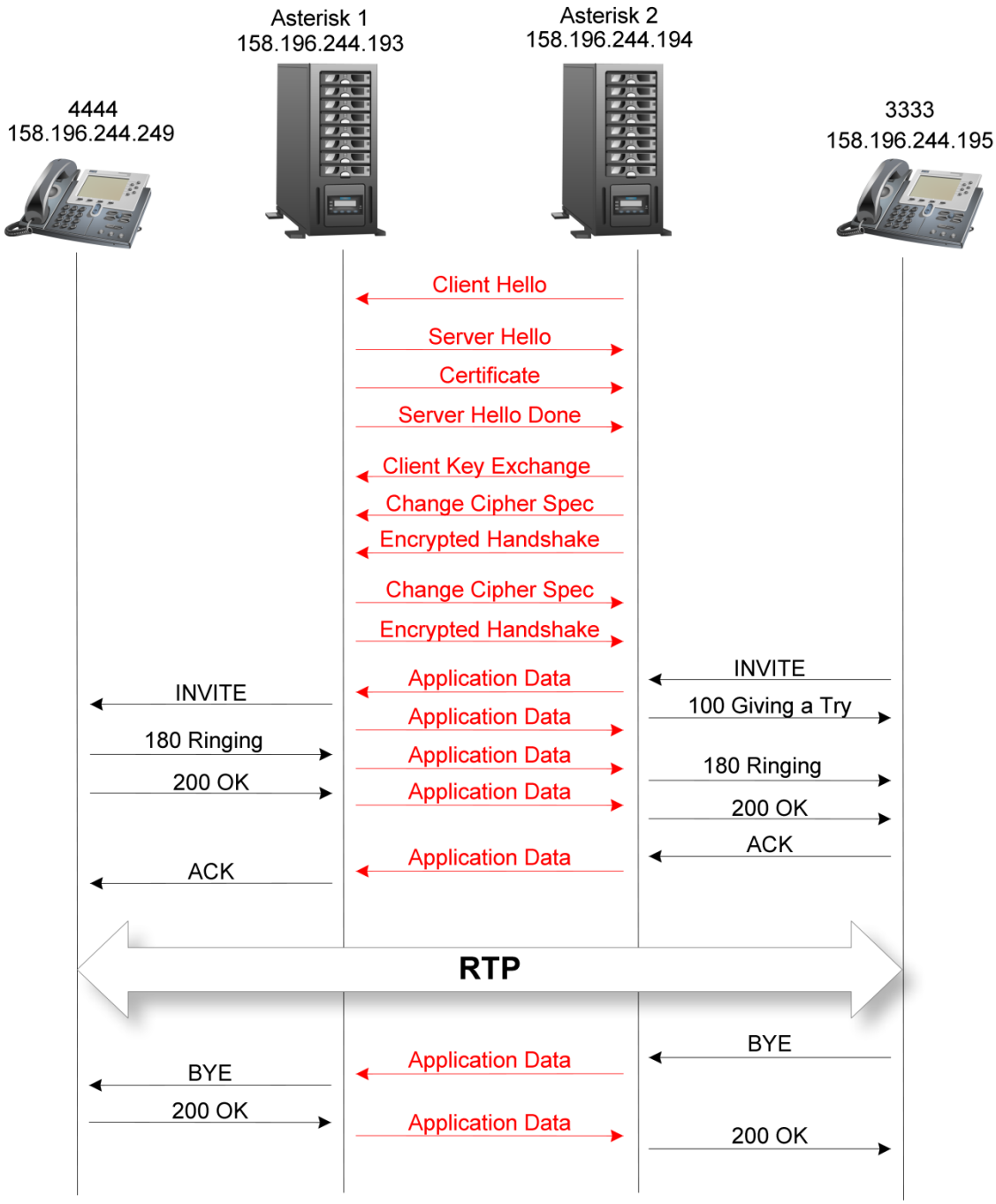
Nyní přistoupíme ke konfiguraci dialplánu neboli konfigurace souboru `extensions.conf`.

```
gedit /etc/asterisk/extensions.conf
[server-2]
exten => 3333,1,Dial(SIP/petr)
exten => _4.,1,Set(CALLERID(num)=${CALLERID(num)})
exten => _4.,2,Dial(SIP/trunk-2/${EXTEN})
```

Příloha VI.: Průběh Handshake protokolu u OpenSIPSu



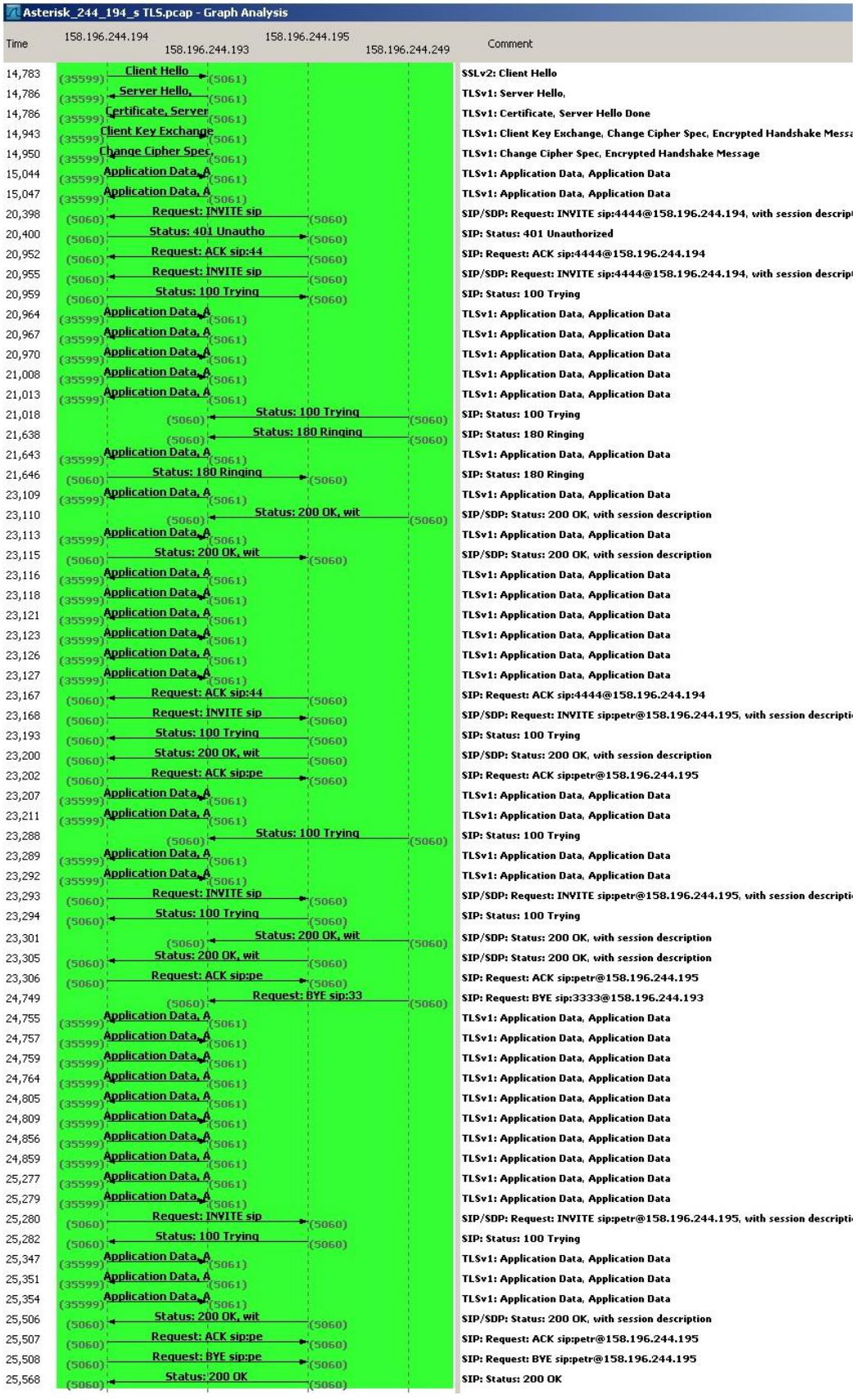
Příloha VII.: Průběh Handshake protokolu u Asterisku



Příloha VIII.: Zabezpečená relace – OpenSIPS

Obr.1:Zachycená zabezpečená relace z programu Wireshark ze 158.196.244.206

Příloha IX.: Zabezpečená relace - Asterisk



Obr.1:Zachycená zabezpečená relace z programu Wireshark z 158.196.244.195

Příloha X.: Popis SIP serverů

Stateless a Stateful Proxy servery

Stateless

Stateless SIP Proxy neboli bezstavové SIP Proxy jsou poměrně jednoduché servery, které pouze přeposílají zprávy, aniž by zkoumaly vzájemné vazby mezi přeposílanými zprávami. Stateless zajišťují správnou návaznost a význam zpráv v signalizaci, ale neumí kontrolovat výměnu zpráv z pohledu smysluplnosti, opakovatelnosti zpráv a hůře detekují nekonečné smyčky. Hlavní nevýhodou oproti Stateful serverům je, že neumí větvení a přesměrování zpráv. Jejich výhoda spočívá v jednoduchosti a tím pádem i rychlosti.

Stateful

Stateful SIP Proxy neboli stavové SIP Proxy jsou oproti Stateless serverům mnohem komplexnější, ale zároveň složitější a pomalejší. Jelikož se jedná o stavový server, tak po přijetí požadavku (např. INVITE) si server vytvoří záznam stavu a „drží“ důležité informace, dokud nedojde k ukončení transakce či dialogu. Z této vlastnosti plyne rozdělení Sateful serverů na:

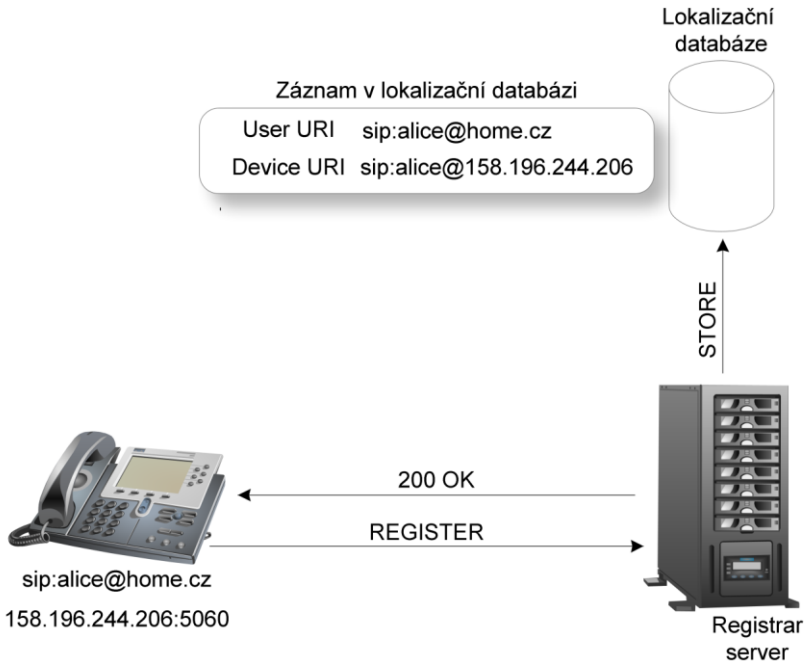
- Transakční – drží stavy k žádosti, dokud není žádost definitivně vyřízena.
- Dialogové – drží stavy dialogu, dokud nedojde k ukončení celého spojení.

Velkou výhodou těchto serverů je schopnost provádět větvení zpráv, nebo možnost zachytit opakující se zprávy.

Registrar server

Registrar server je velmi často jako logická část SIP serveru a nikoli jako samotný server. Registrar slouží k registraci uživatelů na SIP serveru. Přijímá požadavky o registraci, čímž získává informace o aktuální poloze uživatele v síti. Tyto nashromážděné informace ukládá do lokalizační databáze. Podrobnou funkci Registrar serveru si vysvětlíme na následujícím příkladu.

Uživatel Alice posílá požadavek na registraci k SIP serveru (*REGISTER*). Tento požadavek přijme logická část SIP serveru, tedy Registrar server. Zpráva *REGISTER* obsahuje User URI (*sip:alice@home.cz*) a Device URI (*sip:alice@158.196.244.206:5060*). Registrar server si uloží informace od Alice do lokalizační databáze (*STORE*). Pokud registrace proběhla úspěšně, Registrar server vrátí odpověď *200 OK* a registrace je ukončena.

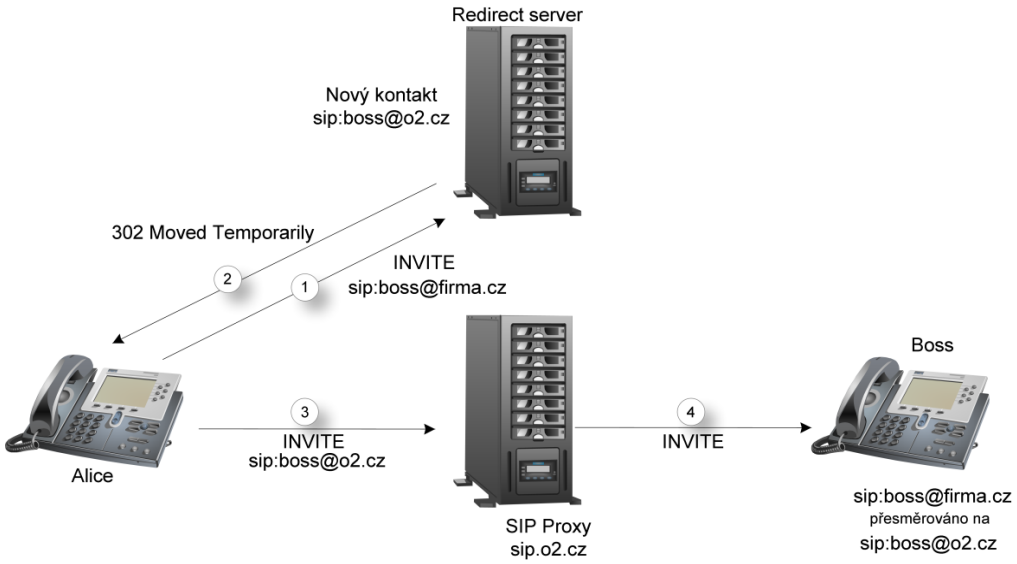


Obr.1: Průběh registrace na Registrar serveru

Redirect server

Redirect server je speciální druh serveru, který slouží k nalezení neboli lokalizaci konkrétního volaného účastníka pro potřeby volajícího. Tento druh serveru přijímá požadavky a odesílá zpět odpověď obsahující lokalizaci konkrétních potenciálně volaných účastníků. Polohu potenciálně volaných účastníků zjišťuje na základě údajů v lokalizační databázi, kterou spravuje Registrar server. Z těchto údajů vytváří seznam aktuálních lokalizací volaného a odesílá je volajícímu v odpovědi typu 3xx. Funkci Redirect serveru si ukážeme na následujícím příkladu.

Volající Alice chce komunikovat se svým šéfem. Jinými slovy, Alice (*alice@firma.cz*) posílá zprávu *INVITE*, kde říká, že chce komunikovat se šéfem (*boss@firma.cz*). *INVITE* od Alice přijme Redirect server, který zjistí, že šéf (*boss*) se již nenachází v doméně *firma.cz*, jelikož je na služební cestě v Praze a tudíž mám nový kontakt (*boss@o2.cz*). Redirect server vrátí Alici odpověď *302 Moved Temporarily*, která obsahuje v poli *contact* SIP URI, na kterou je boss přesměrován. Po přijetí této zprávy, Alice posílá nový *INVITE* na URI z pole *contact*, kterou dostala v odpovědi 302 *Moved Temporarily*. Tento *INVITE* dorazí na SIP Proxy *sip.o2.cz*, která o bossovi ví a přepošle mu příchozí *INVITE* od Alice.



Obr.2: Průběh lokalizaci účastníka pomocí Redirect serveru

Příloha XI.: MGCP žádosti a odpovědi

Žádosti a odpovědi:

Žádosti:

- CRCX (Create Connection) – požaduje vytvoření spojení mezi dvěma koncovými body.
- DLCX (Delete Connection) – ukončuje vytvořené spojení. Ve zprávě ACK jsou posílány statistiky volání.
- MDCX (Modify Connection) – požaduje změny parametrů sestavovaného spojení.
- NTFY (Notify) – MG indikuje CA, že zaregistroval událost, jejíž notifikací si Call Agent předtím vyžádal.
- RQNT (Notification Request) – požadavek na MG(Media Gateway), aby poskytla notifikaci při vyskytnutí konkrétní události u koncového terminál (uživatele).
- AUEP (Audit Endpoint) – monitoruje stav koncového bodu.
- AUCX (Audit Connection) –monitoruje parametry související se spojením.
- RSIP (Restart In Progress) – signalizuje, zda koncový bod je v provozu či mimo provoz.

Odpovědi:

000 – 099	Response acknowledgements (potvrzení odpovědi)
100 – 199	Provisional response (prozatímní odpověď)
200 – 299	Succesful completion (úspěšné dokončení)
400 – 499	Transient error (přechodná chyba)
500 – 599	Permanent error (trvalá chyba)
800 – 899	Package specific response code (specifické kódy)
900 – 999	Reason codes (důvod chyby)

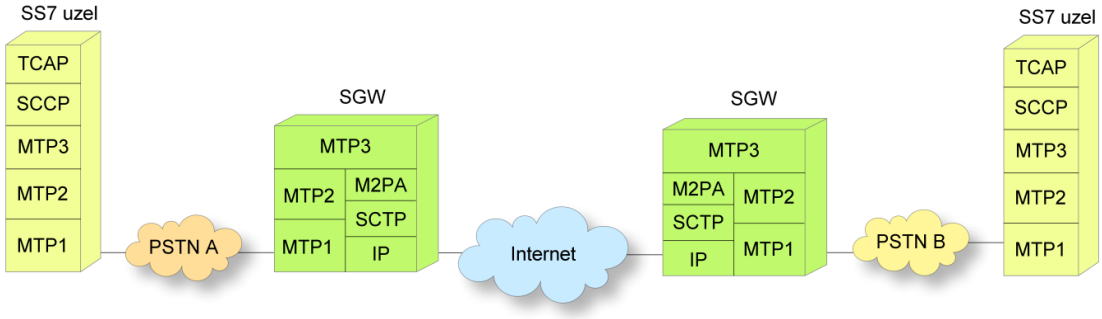
Tab.1: Typy MGCP odpovědí

Příloha XII.: SIGTRAN - uživatelsky adaptační vrstva

Uživatelsky adaptační vrstva

M2PA

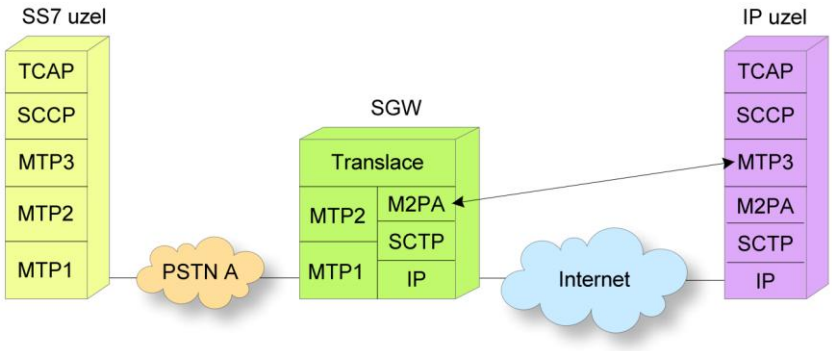
M2PA (MTP2 Peer-to-peer Adaptation layer) je protokol, který přenáší SS7 MTP signalizační zprávy přes IP síť pomocí SCTP protokolu. Je to adaptační protokol mezi MTP3 a SCTP. M2PA zvládá zachovat originální topologii SS7 sítě (všechny síťové elementy jako je např. STP). Jediná věc, která je odlišná, je přenos signalizace přes IP namísto tradiční 64 kbit/s linky.
M2PA může být použit mezi dvěma IP signalizačními uzly v IP síti, nebo mezi SGW (Signaling Gateway) a IP signalizačním uzlem. Nejčastěji se M2PA používá mezi dvěma SGW. Obrázek (*Obr.1*) zobrazuje dvě vzdálené SS7 sítě, které jsou spolu spojeny přes méně nákladnou IP síť. SS7 linky jsou určeny pouze pro signalizaci. Pokud jsou SS7 linky určeny pouze signalizačnímu přenosu, je šířka pásma přiřazena kontinuálně, tudíž občasné využití SS7 linek neúčinně využívá šířku pásma, která je omezena zdrojem. IP řešení mísí signalizační přenos s IP přenosem, čímž redukuje náklady na signalizaci a tím pádem může být linka sdílena více uživateli.



Obr.1: Spojení SS7 sítí přes internet prostřednictvím M2PA

M2UA

M2UA (MTP2-User Adaptation layer) také adaptuje MTP3 do SCTP. Je to protokol, který posílá signalizační zprávy mezi MTP3 vrstvou na MGC (Media Gateway Controller) a MTP2 vrstvou na SGW. Tento protokol je používán hlavně pro komunikaci typu klient – server, např. SG – SP.
Komunikace mezi SG a SP je zobrazen na následujícím obrázku. Protokol MTP3 běžící na signalizačním bodě (SP) využívá MTP2 na signalizační bráně (SGW). Jelikož SP obsahuje i M2UA, tak MTP3 nezjistí, že využívá MTP2 vzdáleně, tedy na SGW místo u sebe na SP. Tento proces se nazývá *backhauling*.

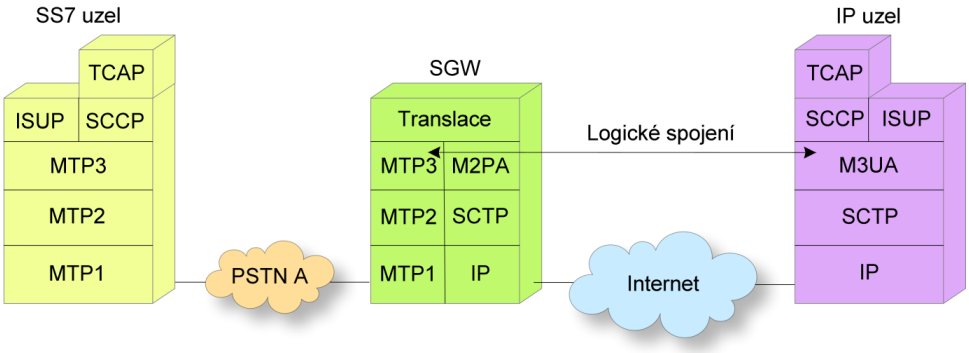


Obr.2: Využití procesu backhauling u M2UA

M2UA je občas využíván při nízké hustotě fyzické SS7 linky v některé části sítě, nebo pokud SGW jsou ve velké vzdálenosti od sebe. V těchto případech *backhauling* může spojit několik takový SP do jednoho centralizovaného elementu sítě, tudíž dovoluje těmto vzdáleným uzlům sdílet jednu SGW.

M3UA

M3UA (MTP3-User Adaptation layer) pracuje na bázi klient-server, právě jako M2UA. Poskytuje vzdálené spojení mezi dvěma SS7 vrstvami v SGW a MGC (IP uzel). Na obrázku (*Obr.3*) můžeme vidět, že SGW má MTP3 vrstvu, která komunikuje s ISUP/SCCP vrstvou na MGC (IP uzlu). MTP3 v SGW neví, že se jedná o vzdáleného uživatele a podobně ISUP/SCCP vrstva v IP uzlu neví, že MTP3 vrstva je od SGW a ne jeho. Toto je další příklad *backhauling*u.



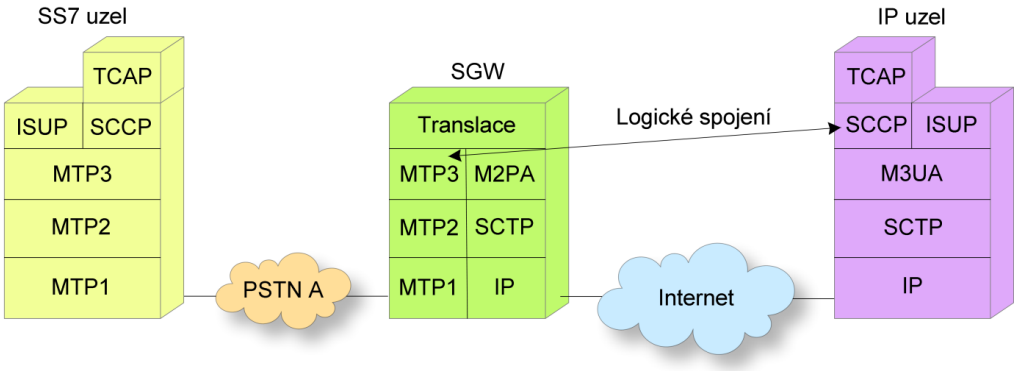
Obr.3: Využití procesu backhauling u M3UA

Pokud je M3UA využíván v all-IP síti, ve které se nenachází jen SS7 uzly, nahradí MTP3 vrstva oba IP uzly a pracuje v point-to-point režimu, který se nazývá jako IP signalizační bod (IP Signaling Point). M3UA je jeden z protokolů uživatelsky adaptační vrstvy, který odstraňuje SS7 vrstvy ze signalizačních bodů a mění topologii sítě na více typickou IP síť. Tento systém může lépe využívat efektivity IP sítě a levnější infrastrukturu a může využívat velkou šířku pásma dostupnou přes IP síť. Flexibilita a schopnost lepšího využití IP sítě umožňuje, aby se M3UA stal protokolem pro UMTS síť.

SUA

Při přechodu z SS7 sítě na IP síť chtěli operátoři zachovat podporu některých cenných aplikací z tradiční telefonní sítě, jako jsou bezplatnost, předplatné či roaming.

Pracovní skupina SIGTRAN vyhověla jejich požadavku a definovala tzv. SUA (SCCP User Adaptation) vrstvu. SUA poskytuje tyto služby nejenom v IP síti. SUA umožňuje efektivněji využívat IP směrování a šířku pásma. IP uzly s SUA jsou jednodušší a proto levnější, než ostatní adaptační vrstvy uzlů.



Obr.4: Využití procesu backhauling u SUA

Hlavními úkoly SUA vrstvy jsou přenosy SCCP uživatelských dat mezi SGW a MGC a mapování mezi SCCP adresami a IP adresami v SGW. SUA neumožňuje přenos ISUP zpráv. Z tohoto důvodu vybralo 3GPP jako standardní signalizační protokol pro centrální části UMTS sítě M3UA, zatímco SUA slouží jako doplněk pro uzly s databázemi např. HLR (Home Location Register).